UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Corso di Laurea Magistrale in Ingegneria dell'Automazione

Strategie di controllo per reti Booleane

Laureanda: Nicoletta Bof

Relatore: Prof. Ettore Fornasini

Controrelatore: Prof.a Maria Elena Valcher

Anno accademico 2013 - 2014
Padova, 10 dicembre 2013

Indice

Introduzione

L'argomento trattato in questa tesi riguarda le reti Booleane, ovvero i sistemi dinamici che coinvolgono variabili Booleane che ad ogni istante interagiscono tra di loro e vanno a determinare, attraverso funzioni logiche, il nuovo valore delle variabili stesse all'istante successivo. Queste reti possono essere prive di variabili di ingresso, e quindi possono solo evolvere in libera, oppure possono presentare anche delle variabili di ingresso, sempre di tipo Booleano; in quest'ultimo caso le reti Booleane vengono dette di controllo.
Visto che le variabili in gioco assumono solo due valori, 0 o 1, le configurazioni possibili delle variabili sono in numero finito, per cui una rete Booleana è un sistema a stati finiti, il cui funzionamento, per lo meno nel caso di un numero piccolo di variabili, è facilmente rappresentabile attraverso un grafo.
L'interesse per questo tipo di reti nasce in campo biologico con un lavoro di Kauffman del 1962, [11], nel quale si propone una modellizzazione dei geni come variabili Booleane (aventi il valore 1 se il gene è attivo o il valore 0 se il gene è inattivo); nello stesso lavoro, Kauffman studia il comportamento di reti Booleane generali, costruite in maniera aleatoria, ed in particolare ne studia i cicli e la stabilità, trovando delle analogie con il funzionamento a livello cellulare dei geni. Per lungo tempo i modelli Booleani non sono stati presi in considerazione come un reale metodo di modellizzazione dei sistemi biologici, e sono stati di gran lunga privilegiati i sistemi di equazioni differenziali. Nei primi anni 2000, invece, compaiono modelli matematici di sistemi biologici ottenuti con reti Booleane: un modello per la regolazione dei geni negli embrioni di *Drosophila Melanogaster*, [24], dei modelli per il ciclo di vita di diversi lieviti, [22], [23], modelli della rete di regolazione dei geni per lo sviluppo dei fiori e altri modelli legati al mondo vegetale, [25] e [26], un modello del ciclo di vita delle cellule dei mammiferi, [20], e modelli del funzionamento dell'operone *lac* nel batterio *Escherichia coli*, [12].
La ricerca non si è unicamente concentrata sull'uso di queste reti per modellare sistemi biologici, ma si è anche sviluppata dal punto di vista teorico, dello studio e delle proprietà di questi sistemi. Soprattutto dopo l'introduzione di una tecnica per individuare una forma algebrica per questo tipo di reti, [1], sono stati numerosi gli studi legati a questo argomento, e sono andati ad investigare i diversi aspetti che riguardano la teoria dei sistemi: comportamento asintotico, raggiungibilità, controllabilità, stabilizzabilità, [4], osservabilità, [3], retroazione dallo stato, [5], controllo ottimo,[7],[9],[10], e altri ancora.

La tesi si concentra sulla ricerca di strategie per il controllo dall'uscita di queste reti, argomento riguardo al quale in letteratura esiste un solo lavoro, [6]; le strategie elaborate vengono poi applicate ad uno dei modelli biologici prima citati, quello del funzionamento dell'operone *lac*.

La tesi si sviluppa come segue:

- Nel primo capitolo vengono riportati alcuni preliminari matematici che risultano utili per la comprensione dell'argomento trattato: essi riguardano la logica Booleana, il prodotto semi-tensoriale e la scrittura in forma algebrica di funzioni logiche, i sistemi positivi, le catene di Markov e alcune nozioni sui grafi.

- Il secondo capitolo tratta le reti Booleane (quindi quelle prive di ingresso), studiandone il comportamento a regime, i cicli e la stabilità della rete stessa.

- Nel terzo capitolo si considera la presenza di ingressi e di uscite nelle reti Booleane, per cui vengono trattati argomenti quali la raggiungibilità e la controllabilità, la stabilizzazione, l'osservabilità e la ricostruibilità.

- Il quarto capitolo si occupa in maniera approfondita delle strategie di controllo per le reti Booleane: la prima strategia trattata è la retroazione dallo stato, seguita dalla retroazione dall'uscita ed infine dal controllo ottimo.

- Nel quinto capitolo si studia il funzionamento dell'operone *lac*, se ne ottiene un modello tramite una rete Booleana e a questo modello si applicano le strategie di controllo dall'uscita elaborate nel quarto capitolo.

1 Preliminari matematici

I sistemi a stati Booleani, come verrà meglio esposto nel capitolo successivo, sono sistemi in cui le variabili di stato e di ingresso possono assumere due soli valori, 0 e 1. Questo comporta un notevole legame con il mondo matematico della logica binaria, in cui variabili che possono assumere solo due valori sono l'argomento di funzioni che a loro volta danno come risultato solo due possibili valori. Inoltre, come si vedrà in seguito, le reti Booleane possono essere descritte da sistemi con matrici di evoluzione di stato i cui elementi assumono due soli valori, 0 e 1, e quindi da sistemi positivi assai particolari che presentano anche un forte legame con le catene di Markov.
Nel seguito del capitolo sono riportati concetti di base di logica Booleana, un metodo per trovare la forma algebrica di un sistema di equazioni logiche,alcuni risultati riguardanti i sistemi positivi e le catene di Markov. L'ultima paragrafo è dedicato alla teoria dei grafi.

1.1 Logica Booleana

Nella logica Booleana l'insieme su cui si lavora è quello composto dagli elementi 0 e 1, che nel seguito viene denominato con $\mathcal{B} = \{1, 0\}$.

Definizione 1.1: Una variabile α si dice *logica* se i suoi valori appartengono a $\mathcal{B}$. Una variabile logica è una *costante* se assume un valore fisso, 0 o 1.

Date due variabili logiche α e β, i connettori logici esprimono le operazioni che possono essere effettuate con queste. I tre più importanti connettori logici sono:

- NOT: esprime l'operazione unaria di negazione e si indica con il simbolo $\neg$; in particolare $\neg\alpha = 1$ se e solo se $\alpha = 0$;
- AND: esprime l'operazione binaria di congiunzione tra α e β, e si indica con il simbolo $\wedge$; $\alpha \wedge \beta = 1$ se e solo se sia α che β sono pari a 1;
- OR: esprime l'operazione binaria di disgiunzione, e si indica con $\vee$; $\alpha \vee \beta = 0$ se e solo se sia α che β sono uguali a 0.

Si può dimostrare che tutte le altre operazioni effettuabili tra 2 variabili logiche, e quindi tutti gli altri connettori esistenti, possono essere espresse tramite queste 3 operazioni fondamentali.

Definizione 1.2: Una *funzione logica* f è una espressione logica che, attraverso i connettori logici, coinvolge n variabili logiche e eventualmente anche alcune costanti. È quindi una mappa $f : \mathcal{B}^n \to \mathcal{B}$.

Osservazione 1.1: Tra i connettori logici, la negazione ha sempre la precedenza sugli altri; per modificare la precedenza tra connettori si usano le parentesi, con la regola che le operazioni tra parentesi vanno svolte per prime. □

Le funzioni logiche possono essere rappresentate tramite tabelle di verità che, a seconda del valore assunto dalle variabili logiche, indicano qual è il risultato della funzione.

Esempio 1.1: In Tabella 1.1 si riportano le "tabelle di verità" dei 3 connettori logici precedentemente richiamati, e anche quelle di altri 3 connettori molto usati, l'implicazione logica, $\to$, la doppia implicazione logica, $\leftrightarrow$, e la disgiunzione esclusiva o XOR, $\bar{\vee}$.

α	β	$\neg\alpha$	$\alpha \wedge \beta$	$\alpha \vee \beta$	$\alpha \to \beta$	$\alpha \leftrightarrow \beta$	$\alpha \bar{\vee} \beta$
1	1	0	1	1	1	1	0
1	0	0	0	1	0	0	1
0	1	1	0	1	1	0	1
0	0	1	0	0	1	1	0

Tabella 1.1: Tabella di verità dei principali connettori logici

♣

Esempio 1.2: Data l'espressione

$$\delta = (\alpha \wedge \neg\beta) \vee (\gamma \wedge 1) \vee (\alpha \wedge \gamma) \ ,$$

questa è una funzione logica f di 3 variabili logiche, $f : \mathcal{B}^3 \to \mathcal{B}$, la cui tabella di verità è riportata nella Tabella 1.2.

♣

A questo punto risulta importante trovare un modo per esprimere le funzioni logiche sotto forma algebrica di prodotto fra matrici e vettori, dal momento che questo permetterà di rappresentare in forma algebrica le reti Booleane (BN, Boolean networks). Per fare questo è necessario introdurre un tipo di prodotto tra matrici che non è quello usualmente adottato in algebra, ovvero il prodotto semitensoriale.

α	β	γ	$\delta = F(\alpha, \beta, \gamma)$
1	1	1	1
1	1	0	0
1	0	1	1
1	0	0	1
0	1	1	1
0	1	0	0
0	0	1	1
0	0	0	0

Tabella 1.2: Tabella di verità della funzione F.

1.2 Prodotto Semitensoriale

Il prodotto semitensoriale è l'estensione dell'usuale prodotto tra matrici nel caso queste non abbiano le dimensioni compatibili. Questo prodotto permette infatti di calcolare AB, $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{p\times q}$, anche quando n è diverso da p. Per definirlo conviene fornire inizialmente la definizione di prodotto di Kronecker, un altro tipo di prodotto fra matrici.

Definizione 1.3: Il *prodotto di Kronecker* tra 2 matrici, $A \in \mathbb{R}^{m\times n}$ e $B \in \mathbb{R}^{p\times q}$, si indica con $\otimes$ e si definisce nel seguente modo:

$$C = A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & \cdots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{bmatrix}$$

dove a_{ij} indica l'elemento in posizione (i, j) della matrice A, e $a_{ij}B$ indica il prodotto della matrice B per lo scalare a_{ij}. La matrice C appartiene ovviamente a $\mathbb{R}^{mp\times nq}$.

Sfruttando quest'ultima definizione è possibile introdurre il prodotto semitensoriale come segue:

Definizione 1.4: Il *prodotto semitensoriale (sinistro)* delle matrici $A \in \mathbb{R}^{m\times n}$ e $B \in \mathbb{R}^{p\times q}$, si indica con $A \ltimes B$ ed è definito nel seguente modo:

$$A \ltimes B = (A \otimes I_{T/n})(B \otimes I_{T/p})$$

con $T = \text{mcm}(n, p)$, il minimo comune multiplo tra n e p.

Osservazione 1.2: Se $n = p$, il prodotto semitensoriale $A \ltimes B$ diventa l'usuale prodotto tra matrici, $A \ltimes B = AB$. □

Esempio 1.3: Dati i due vettori colonna di dimensione 2, $A = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$, $B = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$, si ha

$$A \ltimes B = (A \otimes I_2)B = \begin{bmatrix} a_1 & 0 \\ 0 & a_1 \\ a_2 & 0 \\ 0 & a_2 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} a_1 b_1 \\ a_1 b_2 \\ a_2 b_1 \\ a_2 b_2 \end{bmatrix} = A \otimes B$$

♣

Il risultato di questo esempio ha validità generale e può essere riassunto nella seguente

Proposizione 1.1 *Dati 2 vettori colonna $X \in \mathbb{R}^m$, $Y \in \mathbb{R}^p$, $Z = X \ltimes Y$ è un vettore colonna di dimensione mp e si ha $Z = X \otimes Y$.*

Nel seguito si riportano altre proprietà del prodotto semitensoriale.

Proposizione 1.2 *Il prodotto semitensoriale gode delle seguenti proprietà*

- *proprietà associativa:* $(A \ltimes B) \ltimes C = A \ltimes (B \ltimes C)$*;*
- *proprietà distributiva:* $(A + B) \ltimes C = (A \ltimes C) + (B \ltimes C)$
  $A \ltimes (B + C) = A \ltimes B + A \ltimes C$;
- $(A \ltimes B)^T = B^T \ltimes A^T$*;*
- *se* $M \in \mathbb{R}^{m \times pn}$*, allora* $M \ltimes I_n = M$*;*
- *se* $M \in \mathbb{R}^{pm \times n}$*, allora* $I_m \ltimes M = M$*.*

Osservazione 1.3: Esiste anche una versione destra del prodotto semitensoriale, che presenta proprietà molto simili a quelle del prodotto semitensoriale sinistro. Nel seguito non verrà utilizzato e quindi non viene qui riportato. □

A questo punto è possibile cercare una forma algebrica con cui esprimere una qualsiasi funzione logica.

1.3 Forma algebrica di funzioni logiche

Introducendo la notazione δ_k^i ad indicare la i-esima colonna della matrice I_k, e Δ_k ad indicare l'insieme formato da tutte le colonne della matrice I_k, si ha che una variabile logica a valori in $\mathcal{B}$ presenta una "forma vettoriale" a valori in Δ_2. In particolare, se $\alpha \in \mathcal{B}$, la sua forma vettoriale $a \in \Delta_2$ è tale per cui

$$a = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \delta_2^1 \;\Leftrightarrow\; \alpha = 1 \;,\; a = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \delta_2^2 \;\Leftrightarrow\; \alpha = 0 \;.$$

Definizione 1.5: Una matrice $L \in \mathbb{R}^{n\times m}$ si dice *logica* se l'insieme formato dalle sue colonne, denotato con $\mathrm{Col}(L)$, è contenuto in Δ_n. L'insieme di tutte le matrici logiche di dimensione $n \times m$ viene denotato con $\mathcal{L}^{n\times m}$.

Definizione 1.6: Dato l'operatore logico $f(p_1, p_2, \ldots, p_n)$, con p_i variabili logiche espresse in forma vettoriale, la matrice M_f, di dimensione 2×2^n, si dice *matrice di struttura* dell'operatore f se

$$f(p_1, p_2, \ldots, p_n) = M_f \ltimes p_1 \ltimes p_2 \ltimes \cdots \ltimes p_n \;.$$

Risulta quindi necessario capire come trovare una matrice di struttura di un dato operatore logico. Sfruttando le prossime proposizioni si ricaverà un procedimento per ottenerla, e si dimostrerà inoltre che la matrice di struttura è unica ed è logica.
Per prima cosa viene analizzato il prodotto semitensoriale delle variabili Booleane nella loro forma vettoriale, come compare nella Definizione 1.6, cioè $x = p_1 \ltimes p_2 \ltimes \cdots \ltimes p_r$.
Esiste una corrispondenza biunivoca tra x e i diversi p_i che intervengono nel prodotto, come stabilisce il seguente

Lemma 1.1 *Siano $p_i,\ i = 1, 2, \ldots, n$, variabili logiche in forma vettoriale, cioè $p_i \in \Delta_2$. Se $x = p_1 \ltimes p_2 \ltimes \cdots \ltimes p_n = \ltimes_{i=1}^n p_i$, allora i $p_i,\ i = 1, 2, \ldots, n$, sono univocamente determinati da x.*

Dimostrazione: La dimostrazione del lemma consiste nella ricostruzione dei diversi p_i dato x. Dal momento che $p_i \in \Delta_2$, allora $x \in \Delta_{2^n}$, e si può suddividere x in due vettori di uguale dimensione:

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \;.$$

Poiché x è un vettore canonico si possono verificare due casi:

1. $x_1 \in \Delta_{2^{n-1}}$ e x_2 è pari al vettore nullo di dimensione 2^{n-1};

2. x_1 è pari al vettore nullo di dimensione 2^{n-1} e $x_2 \in \Delta_{2^{n-1}}$.

Visto che il prodotto semitensoriale tra vettori coincide con il prodotto di Kronecker, è facile convincersi che se $x_2 = \mathbf{0}$ allora $p_1 = \delta_2^1$, mentre se $x_1 = \mathbf{0}$ allora $p_1 = \delta_2^2$. In questo modo p_1 è stato univocamente determinato e il procedimento può essere reiterato utilizzando il vettore x_i, con i pari a 1 o 2, che risulta diverso da zero, determinando consecutivamente tutti gli n p_i. ♢

Vista l'importanza del rapporto tra x e i diversi p_i, si riporta nel seguito un metodo che si basa sul Lemma precedente e permette di passare da una forma all'altra.
Dato $x = \delta_{2^n}^i = \ltimes_{j=1}^n p_j$, dove ciascun vettore $p_j \in \Delta_2$ corrisponde ad un particolare valore della variabile logica (Booleana) ρ_j, allora

- i valori di ρ_j possono essere calcolati in maniera induttiva fissando inizialmente $q_0 = 2^n - i$, e calcolando ρ_j e q_j, $j = 1, 2, \ldots, n$, usando ripetutamente la seguente regola[1]

$$\begin{cases} \rho_j = \left\lfloor \frac{q_{j-1}}{2^{n-j}} \right\rfloor \\ q_j = q_{j-1} - 2^{n-j}\rho_j \end{cases} \tag{1.1}$$

- il vettore canonico x, cioè l'indice i, può invece essere calcolato a partire dai valori dei diversi ρ_j, usando:

$$i = \sum_{j=1}^{n} (1 - \rho_j) 2^{n-j} + 1 \tag{1.2}$$

Esempio 1.4: Si consideri $x = p_1 \ltimes p_2 \ltimes p_3 \ltimes p_4$. Si calcoli il valore di x quando $p_1 = p_4 = \delta_2^1$ e $p_2 = p_3 = \delta_2^2$, e, viceversa, il valore delle variabili logiche ρ_i, e quindi dei vettori p_i, $i = 1, \ldots, 4$, quando $x = \delta_{16}^{13}$.
Nel primo caso si utilizza la formula (1.2): i valori scalari dei vettori logici sono $\rho_1 = \rho_4 = 1$, $\rho_2 = \rho_3 = 0$, quindi applicando la formula si ottiene $x = \delta_{16}^7$.
Nel secondo caso per trovare i valori scalari dei p_i, bisogna inizializzare $q_0 = 16 - 13 = 3$, e utilizzare ricorsivamente la formula (1.1); si ottiene quindi $\rho_1 = \rho_2 = 0$, $\rho_3 = \rho_4 = 1$, e i rispettivi valori vettoriali sono $p_1 = p_2 = \delta_2^1$ e $p_3 = p_4 = \delta_2^2$. ♣

[1] $\lfloor a \rfloor$ indica la parte intera di a.

Avendo allora a disposizione la tabella di verità di un qualsiasi operatore logico $f(p_1, \ldots, p_n),\ p_i \in \Delta_2$, ad ogni riga della tabella corrisponde una e una sola combinazione delle variabili p_i, e quindi uno e un solo valore del vettore $x = \ltimes_{i=1}^{n} p_i$. Si ottiene quindi con facilità la tabella di verità relativa ai valori assunti da x.
Ottenuta la tabella di verità relativa alla variabile x, trovare la matrice di struttura risulta immediato; se l'operatore logico in questione è $f(p_1, \ldots, p_n),\ p_i \in \Delta_2$, si ha $f(p_1, \ldots, p_n) = M_f \ltimes (\ltimes_{i=1}^{r} p_i) = M_f \ltimes x$. Dal momento che x è un vettore canonico, $x = \delta_{2^n}^{i},\ i = 1, \ldots, 2^n$, $M_f \ltimes x$ corrisponde alla colonna i-esima di M_f. Questo comporta che se a $x = \delta_{2^n}^{i}$ corrisponde 1 nella tabella di verità, la colonna i-esima di M_f sarà δ_2^1, mentre se ci corrisponde 0, la colonna i-esima sarà δ_2^2 (il risultato di $f(p_1, \ldots, p_n)$ viene fornito in forma vettoriale).

Esempio 1.5: Partendo dall'Esempio 2.2, è facile ricavare sia la tabella di verità riferita a $x = \alpha \ltimes \beta \ltimes \gamma$ che la matrice di struttura di $F(\alpha, \beta, \gamma)$, utilizzando le considerazioni precedenti:

x	$\delta = F(\alpha, \beta, \gamma)$
δ_8^1	1
δ_8^2	0
δ_8^3	1
δ_8^4	1
δ_8^5	1
δ_8^6	0
δ_8^7	1
δ_8^8	0

$$M_F = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

♣

Dal procedimento con cui si ricava la tabella di verità e la matrice di struttura, è facile dimostrare il seguente

Teorema 1.1 *Data una funzione logica $f(p_1, \ldots, p_n)$, con $p_i \in \Delta_2,\ i = 1, \ldots, n$, variabili logiche, esiste una sola matrice M_f di dimesioni 2×2^n tale che*

$$f(p_1, \ldots, p_n) = M_f \ltimes p_1 \ltimes \cdots \ltimes p_n \ .$$

M_f è la matrice di struttura dell'operatore f e $M_f \in \mathcal{L}^{2 \times 2^n}$.

Dimostrazione: Dato l'operatore logico $f(p_1, \ldots, p_n)$ è sempre possibile costruirne la tabella di verità riferita al prodotto $x = p_1 \ltimes \cdots \ltimes p_n$, e da questa si può costruire la matrice di struttura M_f, che presenta come colonna i-esima il valore logico (in forma vettoriale) assunto da

f in corrispondenza di $x = \delta_{2^r}^i$. L'esistenza è stata così provata. Per dimostrare l'unicità si può procedere per assurdo assumendo che esistano 2 matrici di struttura distinte $\bar{M}_f$ e $\tilde{M}_f$. Se sono distinte, vuol dire che hanno almeno una colonna distinta, per esempio la i-esima, ma allora quando $x = \delta_{2^r}^i$, $\bar{M}_f x \neq \tilde{M}_f x$ e questo è assurdo perché sono matrici di struttura dello stesso operatore logico. Infine la matrice che si ottiene è una matrice logica perché l'operatore restituisce un valore logico che è rappresentato vettorialmente da δ_2^1 o δ_2^2, e quindi le colonne di M_f sono vettori canonici. $\diamondsuit$

Con il procedimento appena esposto si è visto come trovare la forma matriciale di un operatore logico; a questo punto si può applicare la tecnica per trovare la rappresentazione matriciale di sistemi di equazioni logiche.

1.4 Forma algebrica di sistemi di equazioni logiche

Definizione 1.7: Un *sistema di equazioni logiche* è un sistema

$$\begin{cases} f_1(\rho_1, \ldots, \rho_n) = \gamma_1 \\ f_2(\rho_1, \ldots, \rho_n) = \gamma_2 \\ \quad \vdots \\ f_m(\rho_1, \ldots, \rho_n) = \gamma_m \end{cases} \tag{1.3}$$

dove $f_i,\ i = 1, \ldots, m$, sono funzioni logiche, $\rho_j,\ j = 1, \ldots, n$, sono variabili logiche e $\gamma_h,\ h = 1, \ldots, m$, sono costanti logiche note.
Un insieme di costanti logiche $\chi_j,\ j = 1, \ldots, n$ tale che $\rho_j = \chi_j,\ j = 1, \ldots, n$, soddisfa (1.3), si dice *soluzione* del sistema (1.3).

Ogni singola equazione del sistema può essere espressa in forma matriciale sfruttando il procedimento esposto nel paragrafo precedente. Considerando p_i la forma vettoriale corrispondente alla variabile logica ρ_i, $i = 1, \ldots, n$, c_j la forma vettoriale corrispondente a $\gamma_j,\ j = 1, \ldots, m$, e $x = \ltimes_{i=1}^n p_i$ si ottiene:

$$\begin{cases} M_1 x = c_1 \\ M_2 x = c_2 \\ \quad \vdots \\ M_m x = c_m \end{cases} \tag{1.4}$$

dove $M_i,\ i = 1, \ldots, m$, è la matrice di struttura relativa all'operatore logico f_i. A questo punto si possono moltiplicare membro a membro le diverse equazioni del sistema, tramite il prodotto semitensoriale; detto

$w = \ltimes_{i=1}^{m} c_i \ \in \Delta_{2^m}$, il suo valore è in rapporto biunivoco con i valori assunti dai singoli c_i, secondo il Lemma 1.1. Si ottiene

$$w = M_1 x \ltimes M_2 x \ltimes \cdots \ltimes M_m x \ .$$

All'espressione a destra dell'uguaglianza si può associare una tabella di verità che ad ogni valore della $x \in \Delta_{2^n}$ associa un unico valore appartenente a Δ_{2^m}. L'espressione a destra si può quindi sostituire con il prodotto di una nuova matrice $\bar{M} \in \mathcal{L}^{2^m \times 2^n}$ per il vettore x; $\bar{M}$ ha come i-esima colonna il vettore canonico appartenente a Δ_{2^m} che corrisponde nella tabella di verità al valore associato a $x = \delta_{2^n}^i$.
Il sistema di equazioni logiche di partenza (1.3) si può quindi riscrivere come

$$w = \bar{M} x \ . \tag{1.5}$$

La ricerca di una (o di tutte) le soluzioni del sistema risulta a questo punto immediata: si ha infatti che la soluzione del sistema esiste se e solo se esiste i tale che $\mathrm{Col}_i(\bar{M}) = w$, e in particolare tutte le soluzioni si trovano grazie al seguente

Teorema 1.2 *Il sistema di equazioni logiche (1.3), riscritto nella forma (1.5), presenta tutte e sole le seguenti soluzioni*

$$x = \delta_{2^n}^{\lambda} \ , \qquad \lambda \in \Lambda = \left\{ \lambda | \mathit{Col}_{\lambda}(\bar{M}) = w \right\} \ .$$

Esempio 1.6: Si consideri il seguente sistema di equazioni logiche:

$$\begin{cases} (\alpha_1 \wedge \alpha_2) \vee \alpha_3 = \gamma_1 \\ \alpha_2 \wedge \alpha_3 = \gamma_2 \\ \alpha_1 \vee \alpha_2 = \gamma_3 \end{cases}$$

Le tabelle di verità relative alle 3 equazioni, anche in funzione della variabile vettoriale $x = a_1 \ltimes a_2 \ltimes a_3$ (con a_i versione vettoriale della variabile logica α_i), sono riportate in Tabella 1.3.

Le matrici di struttura relative alle diverse funzioni logiche sono:

$$M_1 = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \quad M_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

$$M_3 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \ .$$

α_1	α_2	α_3	x	f_1	f_2	f_3
1	1	1	δ_8^1	1	1	1
1	1	0	δ_8^2	1	0	1
1	0	1	δ_8^3	1	0	1
1	0	0	δ_8^4	0	0	1
0	1	1	δ_8^5	1	1	1
0	1	0	δ_8^6	0	0	1
0	0	1	δ_8^7	1	0	0
0	0	0	δ_8^8	0	0	0

Tabella 1.3: Tabella di verità delle funzioni presenti nel sistema di equazioni logiche dato.

Moltiplicando membro a membro, e introducendo $w = c_1 \ltimes c_2 \ltimes c_3$, con c_i versione vettoriale di γ_i, si ottiene l'equazione

$$w = M_1 x \ltimes M_2 x \ltimes M_3 x \ .$$

La tabella di verità associata all'espressione a destra dell'uguaglianza e la relativa matrice sono:

x	$M_1 x \ltimes M_2 x \ltimes M_3 x$
δ_8^1	δ_8^1
δ_8^2	δ_8^3
δ_8^3	δ_8^3
δ_8^4	δ_8^7
δ_8^5	δ_8^1
δ_8^6	δ_8^7
δ_8^7	δ_8^4
δ_8^8	δ_8^8

$$\bar{M} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Il sistema è stato quindi riscritto come

$$w = \bar{M} x \tag{1.6}$$

Se si cerca la soluzione del sistema di partenza quando $\gamma_1 = \gamma_2 = 0$ e $\gamma_3 = 1$, basta calcolare il relativo valore di w che risulta pari a δ_8^7. Analizzando la matrice $\bar{M}$, e richiamando il Teorema 1.2, si trova che le soluzioni del sistema (1.6) sono $x = \delta_8^4$ o $x = \delta_8^6$, a cui corrispondono le variabili logiche di partenza $\alpha_1 = 1,\ \alpha_2 = \alpha_3 = 0$ oppure $\alpha_1 = \alpha_3 = 0,\ \alpha_2 = 1$.
Se invece si cercano le soluzioni del sistema quando $\alpha_1 = 0$ e $\alpha_2 = \alpha_3 = 1$, si ha che $w = \delta_8^5$ e analizzando $\bar{M}$ si nota che non esiste alcuna soluzione

del sistema (come risulta evidente anche confrontando la tabella di verità delle singole funzioni logiche). ♣

Con il procedimento adottato risulta possibile risolvere un sistema di equazioni logiche attraverso la risoluzione di un sistema matriciale in cui il vettore incognito può variare solo tra i vettori canonici. Dal momento che l'evoluzione dello stato nelle reti Booleane è descritta da sistemi di equazioni logiche, questo implica che l'evoluzione può anche essere descritta attraverso l'uso di matrici logiche.
Visto che le matrici logiche sono matrici che presentano tutte le colonne che sono vettori canonici, e che il vettore di partenza è a sua volta canonico, il sistema che si ottiene è un sistema positivo. Oltretutto, poichè la matrice di evoluzione ha le colonne che sono vettori canonici, ed è quindi stocastica per colonne, e il vettore di partenza è un vettore canonico, si ha che la rete Booleana è una catena di Markov in cui invece di lavorare con vettori riga si lavora con vettori colonna (e quindi entra in gioco l'operazione di trasposizione). Nel prossimo paragrafo si trattano brevemente i sistemi positivi e le catene di Markov relativamente agli aspetti che avranno maggiore importanza nel resto del lavoro.

1.5 Sistemi positivi e catene di Markov

Definizione 1.8: Un sistema si dice *positivo* se tutte le variabili di ingresso, stato e uscita sono vincolate ad assumere solo valori non negativi

Dato il seguente modello di stato

$$\begin{cases} x(t+1) = Fx(t) + Gu(t) \\ y(t) = Hx(t) + Du(t) \end{cases} \tag{1.7}$$

con $x(t) \in \mathbb{R}^n,\ u(t) \in \mathbb{R}^m,\ y(t) \in \mathbb{R}^p,\ F \in \mathbb{R}^{n\times n},\ G \in \mathbb{R}^{n\times m},\ H \in \mathbb{R}^{p\times n},\ D \in \mathbb{R}^{p\times m}$, è facile provare la seguente

Proposizione 1.3 *Il modello di stato 1.7 è positivo* $\Leftrightarrow$ *le matrici* $F,\ G,\ H,\ D$ *sono non negative.*

Nel seguito si utilizza la seguente notazione per matrici non negative:

- $M \in \mathbb{R}_+^{p\times m}$ matrice non negativa di dimensione $p \times m$;
- $M \gg 0$ matrice con tutti gli elementi strettamente maggiori di 0;

- $M > 0$ matrice non negativa con almeno un elemento strettamente maggiore di 0;
- $M \geq 0$ matrice non negativa (eventualmente anche la matrice nulla).

Definizione 1.9: Data una matrice $F > 0$ quadrata, questa si dice

- *primitiva* se esiste $h > 0$ tale che $F^h \gg 0$;
- *irriducibile* se $\forall\ r, s = 1, \ldots, n\ \exists\ h\ |[F^h]_{rs} > 0$;
- *riducibile* se $\exists\ r, s = 1, \ldots, n\ |[F^h]_{rs} = 0\ \forall\ h$;
- *strettamente positiva* se $[F]_{rs} > 0\ \forall\ r, s = 1, \ldots, n$.

Nel caso di sistemi positivi, risultano importanti i cambi di base effettuati tramite matrici di permutazione perchè conservano la positività della matrice.
Una permutazione è una mappa biiettiva di n interi in se stessi, che viene indicata nel seguente modo:

$$\sigma = \begin{pmatrix} 1 & 2 & \ldots & n \\ i_1 & i_2 & \ldots & i_n \end{pmatrix} \quad \sigma(k) = i_k\ .$$

A questa permutazione si può associare la matrice di permutazione

$$\Pi_\sigma = \begin{bmatrix} \delta_n^{i_1} & \delta_n^{i_2} & \ldots & \delta_n^{i_n} \end{bmatrix}$$

con δ_n^j j-esimo vettore della base canonica. Π_σ è una matrice di rango pieno ed è ortogonale, quindi ${\Pi_\sigma}^{-1} = {\Pi_\sigma}^T$.
Se si effettua un cambio di base utilizzando una matrice di permutazione, si ottiene che le nuove coordinate coincidono con quelle di partenza, ma sono disposte secondo un diverso ordine. Se infatti $\begin{bmatrix} \chi_1 & \chi_2 & \ldots & \chi_n \end{bmatrix}^T$ sono le coordinate rispetto alla base di partenza, le nuove coordinate $\begin{bmatrix} \hat{\chi}_1 & \hat{\chi}_2 & \ldots & \hat{\chi}_n \end{bmatrix}^T$ risultano essere:

$$\begin{bmatrix} \hat{\chi}_1 \\ \hat{\chi}_2 \\ \vdots \\ \hat{\chi}_n \end{bmatrix} = {\Pi_\sigma}^T \begin{bmatrix} \chi_1 \\ \chi_2 \\ \vdots \\ \chi_n \end{bmatrix} = \begin{bmatrix} (\delta_n^{i_1})^T \\ (\delta_n^{i_2})^T \\ \vdots \\ (\delta_n^{i_n})^T \end{bmatrix} \begin{bmatrix} \chi_1 \\ \chi_2 \\ \vdots \\ \chi_n \end{bmatrix} = \begin{bmatrix} \chi_{i_1} \\ \chi_{i_2} \\ \vdots \\ \chi_{i_n} \end{bmatrix}\ .$$

I cambi di base ottenuti attraverso matrici di permutazione vengono chiamati trasformazioni di cogredienza. Utilizzando trasformazioni di cogredienza, e quindi compiendo un riordino degli stati, si può ottenere la forma *normale* delle matrici riducibili:

Proposizione 1.4 *Data una matrice $F > 0, F \in \mathbb{R}_+^{n\times n}$, $n \geq 2$, riducibile, esiste una matrice di permutazione Π tale che*

$$\tilde{F} = \Pi^T F \Pi = \left[\begin{array}{cccc|ccc} \tilde{F}_{11} & & & & & & \\ \mathbf{0} & \tilde{F}_{22} & & & & \mathbf{0} & \\ \mathbf{0} & \cdots & \ddots & & & & \\ \mathbf{0} & \mathbf{0} & \cdots & \tilde{F}_{hh} & & & \\ \hline \star & \star & \cdots & \star & \tilde{F}_{h+1h+1} & & \\ \star & \star & \cdots & \cdots & \cdots & \ddots & \\ \star & \cdots & \cdots & \cdots & \cdots & \star & \tilde{F}_{kk} \end{array}\right] .$$

I blocchi quadrati $\tilde{F}_{ii}$ sono irriducibili o nulli di dimensione 1, e i blocchi riga contrassegnati con $\star$ presentano almeno un blocco diverso da quello nullo. La matrice ottenuta si dice in forma normale e i blocchi $\tilde{F}_{ii},\ i = 1, \ldots, h$ si dicono blocchi isolati.

La forma normale risulterà importante in relazione alle considerazioni successive sulle catene di Markov.

Una catena di Markov $\mathcal{C}$ è un modello teorico di un sistema a tempo discreto che in ogni momento può trovarsi in un qualche stato di un insieme finito $\mathbf{S} = \{S_1, S_2, \ldots, S_n\}$, e in cui il passaggio dallo stato presente al successivo è dettato da una legge probabilistica. Questa legge gode della proprietà di Markovianità, cioè il passaggio da uno stato al successivo dipende solo dallo stato presente ed è indipendente dalla storia passata del sistema.
Nello studio di queste catene è molto usata la seguente

Definizione 1.10: Un vettore $x = \left[\begin{array}{cccc} x_1 & x_2 & \ldots & x_n \end{array}\right]^T$ non negativo si dice *stocastico* se i suoi elementi sommano a 1, cioè $\sum_{i=1}^{n} x_i = 1$.

Una catena di Markov può essere descritta tramite una matrice di transizione P, che come elemento (i, j) contiene la probabilità di passare dallo stato S_i allo stato S_j e che quindi risulta positiva e stocastica per righe, oppure tramite un grafo orientato, i cui vertici sono gli stati della catena $S_1, \ldots, S_n$, e un arco tra il nodo S_i e S_j è presente se e solo se è positiva la probabilità di transitare dallo stato S_i allo stato S_j.
Se la catena è descritta tramite matrice di transizione, detto $x(t)$ il vettore stocastico n-dimensionale che contiene la distribuzione di probabilità sugli stati della catena all'istante t, l'evoluzione dell'andamento della distribuzione di probabilità è descritto dalla seguente equazione:

$$x^T(t+1) = x^T(t)P\ . \tag{1.8}$$

Il vettore $x(t)$ contiene la distribuzione di probabilità nei diversi stati al tempo t, cioè $x_i(t),\ i = 1, \ldots, n$, è la probabilità che all'istante t la catena di Markov sia nello stato S_i.
L'uso di vettori riga al posto di vettori colonna per descrivere l'evoluzione dello stato è legato alla consuetudine, ma non comporta alcuna modifica sostanziale nello studio di questi sistemi.
Nel seguito si riportano dei risultati sulle catene di Markov che verranno utilizzati in relazione alle reti Booleane.

Definizione 1.11: Lo stato S_k si dice *accessibile* da S_i se è positiva la probabilità che la catena, partendo al tempo $t = 0$ da S_i, visiti S_k in un qualche istante t non negativo.

In termini di matrice di transizione, questo comporta l'esistenza di una potenza $h \geq 0$ tale che $[P^h]_{ik} > 0$.

Definizione 1.12: Due stati S_i e S_k *comunicano* se S_k è accessibile da S_i e S_i è accessibile da S_k.

La relazione di comunicazione è riflessiva, simmetrica e transitiva, quindi induce una partizione dell'insieme $\mathbf{S}$ in *classi di comunicazione*.

Definizione 1.13: Presa $\mathcal{K}$ una classe di comunicazione della catena di Markov $\mathcal{C}$, questa si dice:

- *ergodica* se nessun stato esterno a $\mathcal{K}$ è accessibile da uno stato di $\mathcal{K}$;
- *transitoria* se esistono degli stati esterni a $\mathcal{K}$ che sono accessibili da uno stato di $\mathcal{K}$.

È facile convincersi che se la catena entra in una classe ergodica, ci resterà indefinitivamente, mentre se la catena esce da una classe transitoria non può più entrarci.
Le prossime due proposizioni vengono riportate senza dimostrazione:

Proposizione 1.5 *Ogni catena di Markov possiede almeno una classe ergodica. Se un suo stato S_i non appartiene ad una classe ergodica, esiste almeno un cammino orientato con inizio in S_i che termina in una classe ergodica.*

Proposizione 1.6 *Per $t \to +\infty$ la probabilità che lo stato di una catena di Markov appartenga ad una classe ergodica è pari a 1.*

Quest'ultima proposizione comporta che da qualunque stato iniziale si parta, all'infinito lo stato raggiunto dalla catena appartiene ad una classe ergodica.

Utilizzando un opportuno criterio per la numerazionde degli stati, si può fare in modo che la matrice di transizione P evidenzi le diverse classi di comunicazione e le differenzi tra quelle ergodiche e transitorie. Ovviamente se inizialmente gli stati presentano una numerazione, a questa corrisponde una matrice di transizione P; cambiare la numerazione degli stati implica applicare una trasformazione di cogredienza alla matrice P. La seguente numerazione permette di ottenere la matrice di transizione desiderata:

- tutti gli stati appartenenti a una data classe di comunicazione hanno gli indici consecutivi;
- per primi vanno numerati gli stati appartenenti alle classi ergodiche;
- per secondi vanno numerati gli stati che appartengono a classi di primo livello, cioè quelle classi che hanno connessioni dirette solo con le classi ergodiche;
- successivamente si numerano gli stati che appartengono a classi di secondo livello, cioè quelle che per accedere alle classi ergodiche devono passare al massimo per un'altra classe transitoria e così via.

Con questo riordino degli stati la matrice di transizione presenta la seguente forma:

$$P = \left[\begin{array}{cccc|ccc} \mathcal{K}_1 & & & & & & \\ \mathbf{0} & \mathcal{K}_2 & & & & \mathbf{0} & \\ \mathbf{0} & \cdots & \ddots & & & & \\ \mathbf{0} & \mathbf{0} & \cdots & \mathcal{K}_h & & & \\ \hline \star & \star & \cdots & \star & \mathcal{K}_{h+1} & & \\ \star & \star & \cdots & \cdots & \cdots & \ddots & \\ \star & \cdots & \cdots & \cdots & \cdots & \star & \mathcal{K}_k \end{array}\right].$$

I primi h blocchi diagonali si riferiscono alle classi ergodiche, mentre i restanti alle classi transitorie. Per quanto riguarda i blocchi diagonali di dimensione 1, se sono relativi ad una classe ergodica allora contengono l'elemento 1, altrimenti contengono un elemento non negativo minore di 1; se i blocchi sono di dimensione maggiore di uno, allora sono blocchi irriducibili stocastici (per riga) nel caso di classi ergodiche, oppure irriducibili substocastici, cioè presentano almeno una riga con somma degli elementi strettamente minore di uno (ma comunque non negativa).
Da un confronto con la Proposizione 1.4 segue che la matrice P è in forma normale; portare in forma normale una matrice di transizione di una

catena di Markov permette quindi di evidenziare le sue classi ergodiche, cioè permette di capire verso quali stati converge il sistema a regime.

1.6 Grafi

In quest'ultimo paragrafo vengono ripresi dei concetti di teoria dei grafi che torneranno utili nel seguito.

Definizione 1.14: Un *grafo non orientato* è una coppia $G = (V, E)$ in cui V è un insieme finito ed E una famiglia di coppie non ordinate di elementi di V. Gli elementi di V sono detti *vertici o nodi*, mentre gli elementi di E sono detti *lati*.

Per indicare un elemento di E viene usata la notazione $e = \{i, j\},\ i, j \in V$, che indica la presenza di un lato tra i vertici i e j.

Definizione 1.15: Un *cammino* di G è una sequenza di lati consecutivi $e_1, e_2, \ldots, e_k \in E$, cioè del tipo $e_1 = \{v_1, v_2\},\ e_2 = \{v_2, v_3\},\ \ldots,\ e_k = \{v_k, v_{k+1}\}$.

Il concetto che verrà sfruttato nel capitolo successivo riguarda la seguente

Definizione 1.16: Un vertice $v \in V$ si dice *connesso* al vertice $w \in V$ se esiste un cammino che li collega. Per definizione inoltre ogni vertice è connesso con se stesso.

La relazione di connessione gode della proprietà riflessiva, simmetrica e transitiva, quindi è una relazione di equivalenza; questo comporta che la relazione di connessione induce una partizione dell'insieme dei vertici, e ogni partizione è detta *componente connessa*.

Un altro concetto che torna utile successivamente è quello di ciclo, che però viene introdotto nel caso di grafi diretti.

Definizione 1.17: Un *grafo orientato o diretto* è una coppia $G = (V, A)$, in cui V è un insieme finito di *vertici o nodi* ed A è una famiglia di coppie ordinate di vertici, dette *archi*.

In questo caso per indicare un arco $a \in A$ dal nodo i al nodo j si usa la notazione $a = (i, j),\ i, j \in V$.

Definizione 1.18: Un *cammino orientato* di lunghezza k è una sequenza di archi consecutivi $a_1, a_2, \ldots, a_k$, cioè tali che $a_1 = (v_1, v_2),\ a_2 = (v_2, v_3), \ldots,$
$a_k = (v_k, v_{k+1})$.

Un *ciclo* di lunghezza k è un cammino di lunghezza k in cui il primo e l'ultimo vertice coincidono, cioè $v_{k+1} = v_1$. Il ciclo si dice *semplice* se non visita mai due volte lo stesso vertice, a parte quello di partenza.Un ciclo di lunghezza uno, in cui un arco parte da un vertice e vi ritorna, si dice *self-loop*.

Nel seguito, quando si parla di ciclo, si intende sempre un ciclo semplice.

2 Reti Booleane

In questo capitolo vengono trattate in modo approfondito le reti Booleane (in inglese Boolean networks), sistemi dinamici autonomi (cioè che evolvono in assenza di ingressi), in cui gli stati sono variabili logiche Booleane, la cui evoluzione nel tempo è determinata grazie a funzioni logiche. Nel seguito si utilizzerà l'abbreviazione BN per indicare una rete Booleana.
Dopo aver dato una precisa definizione e un primo metodo di rappresentazione della rete tramite grafo, si andrà a studiare la forma algebrica della BN, alla quale è associato un altro tipo di rappresentazione tramite grafo. Successivamente vengono studiati i punti fissi, i cicli e la stabilità del sistema.

2.1 Definizione matematica e grafo di rete

Definizione 2.1: Una *rete Booleana* è un insieme di variabili, dette "nodi", $\chi_1, \chi_2, \ldots, \chi_n$, che interagiscono simultaneamente tra di loro. Ciascun nodo può assumere solo due valori, 1 o 0, quindi $\chi_i \in \mathcal{B},\ i = 1, \ldots, n$ e l'evoluzione dei nodi della rete può essere descritta con il seguente sistema di equazioni

$$\begin{cases} \chi_1(t+1) = f_1(\chi_1(t), \chi_2(t), \ldots, \chi_n(t)) \\ \chi_2(t+1) = f_2(\chi_1(t), \chi_2(t), \ldots, \chi_n(t)) \\ \qquad\qquad\qquad \vdots \\ \chi_n(t+1) = f_n(\chi_1(t), \chi_2(t), \ldots, \chi_n(t)) \end{cases} \tag{2.1}$$

dove ciascuna $f_i,\ i = 1, 2, \ldots, n$, è una funzione logica, $f_i : \mathcal{B}^n \to \mathcal{B}$.

Esempio 2.1: Il seguente sistema di equazioni logiche definisce una BN con tre nodi, χ_1, χ_2 e χ_3:

$$\begin{cases} \chi_1(t+1) = (\chi_1(t) \wedge \chi_2(t)) \vee \chi_3(t) \\ \chi_2(t+1) = [(\chi_1(t) \leftrightarrow \chi_2(t)) \wedge \chi_3(t)] \vee [\chi_2(t) \wedge \neg(\chi_1(t) \vee \chi_3(t))] \\ \chi_3(t+1) = [(\chi_1(t) \bar{\vee} \chi_2(t)) \wedge \chi_3(t)] \vee [(\neg\chi_1(t) \vee \neg\chi_2(t)) \wedge \neg\chi_3(t)] \end{cases} \tag{2.2}$$

Alle singole funzioni logiche sono associate le tabelle di verità della Tabella 2.1.
Questa rete verrà ripresa negli esempi successivi. ♣

χ_1	χ_2	χ_3	f_1	f_2	f_3
1	1	1	1	1	0
1	1	0	1	0	0
1	0	1	1	0	1
1	0	0	0	0	1
0	1	1	1	0	1
0	1	0	0	1	1
0	0	1	1	1	0
0	0	0	0	0	1

Tabella 2.1: Tabella di verità delle funzioni logiche coinvolte nella BN (2.2).

Definizione 2.2: Il *grafo di rete* $\Sigma = \{\mathcal{N}, \mathcal{E}\}$ della BN (2.1) è un grafo orientato che consiste in un insieme di vertici coincidenti con i nodi della rete, $\mathcal{N} = \{\chi_i | i = 1, \ldots, n\}$, e in un insieme di archi $\mathcal{E} \subset \{\chi_1, \ldots, \chi_n\} \times \{\chi_1, \ldots, \chi_n\}$. Un arco dal nodo χ_i al nodo χ_j esiste se $\chi_j(t+1)$ è influenzato dal nodo $\chi_i(t)$, cioè se il valore di $\chi_i(t)$ concorre a determinare il valore di $\chi_j(t+1)$ nella funzione logica f_j.

Osservazione 2.1: Ogni BN ha un solo grafo di influenza, ma ad un grafo di influenza possono corrispondere diverse BNs. □

Un grafo di rete può essere rappresentato graficamente oppure attraverso una matrice di incidenza $\mathcal{I}$, cioè una matrice quadrata di dimensione n che viene costrita nel seguente modo:

$$\mathcal{I} = (b_{ij}), \text{ con } b_{ij} = \begin{cases} 1, & \text{se } (\chi_i, \chi_j) \in \mathcal{E}\ ; \\ 0, & \text{altrimenti.} \end{cases}$$

Esempio 2.2: Il grafo di rete relativo alla BN dell'Esempio 2.1 è riportato in Figura 2.1.

Per come è fatto il sistema ogni nodo dipende da se stesso e dai rimanenti, quindi in questo caso il grafo da poca informazione; la matrice di incidenza non viene riportata perché è composta da tutti elementi unitari.

Si riporta un'altra rete Booleana, con il relativo grafo di influenza e matrice di incidenza; con questa nuova BN risulta più evidente il significato del grafo.

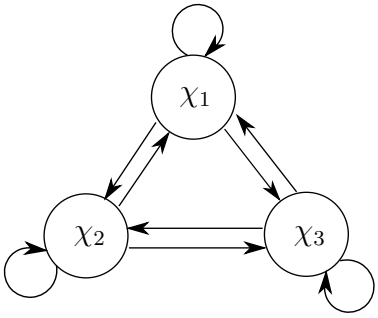
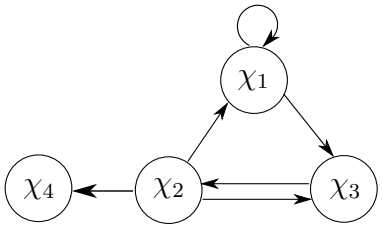


Figura 2.1: Grafo di rete relativo alla BN dell'esempio 3.1.

$$\begin{cases} \chi_1(t+1) = \chi_1(t) \wedge \chi_2(t) \\ \chi_2(t+1) = \neg\chi_3(t) \\ \chi_3(t+1) = \neg\chi_1(t) \vee \chi_2(t) \\ \chi_4(t+1) = \neg\chi_2(t) \end{cases}$$



$$\mathcal{I} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} .$$

♣

2.2 Forma algebrica e grafo di stato

Un approccio molto utile per lo studio delle BN è quello di trovarne la forma algebrica, cioè di riuscire ad esprimerle come un modello di stato lineare discreto.
Per trovare la forma algebrica della BN si procede nel modo indicato nel paragrafo 1.4. Definendo inizialmente $X_i(t),\ i = 1, \dots, n$, forma vettoriale delle variabili logiche $\chi_i(t)$, si può costruire attraverso il prodotto semitensoriale la variabile

$$x(t) = X_1(t) \ltimes X_2(t) \ltimes \cdots \ltimes X_n(t) .$$

Ovviamente $x(t) \in \Delta_{2^n}$ è in rapporto biunivoco con le variabili di cui è il prodotto, cioè con $X_1(t), \dots, X_n(t)$. A questo punto è necessario fare ben attenzione ai diversi tipi di variabili che sono stati introdotti, perchè

risultano simili e facilmente confondibili. Sono state usate tre distinte variabili, tutte strettamente legate fra loro:

- le variabili $\chi_i(t) \in \mathcal{B},\ i = 1, \dots, n$, sono variabili logiche e rappresentano i nodi della BN;
- le variabili $X_i(t) \in \Delta_2,\ i = 1, \dots, n$, sono la rappresentazione vettoriale delle rispettive $\chi_i(t)$, quindi $X_i(t) = \delta_2^1 \Leftrightarrow \chi_i(t) = 1$, oppure $X_i(t) = \delta_2^2 \Leftrightarrow \chi_i(t) = 0$;
- la variabile $x(t) \in \Delta_{2^n}$ contiene tutte le informazioni portate dalle variabili X_i, ed in particolare in ogni istante è in grado di fornire il valore assunto da ogni nodo della rete in quel momento. È la variabile che verrà usata come stato del sistema algebrico associato al sistema di equazioni logiche e può assumere 2^n valori distinti.

Come è già stato evidenziato nel secondo capitolo, è possibile riscrivere ogni singola equazione del sistema (2.1) in forma algebrica, ottenendo

$$X_i(t+1) = M_i x(t) \tag{2.3}$$

in cui M_i rappresenta la matrice di struttura della funzione f_i. L'equazione (2.3) viene chiamata *forma algebrica per l'i-esima componente* della rete. A questo punto, sempre seguendo il procedimento esposto nel capitolo precedente, si moltiplicano membro a membro le diverse forme algebriche per componente, ottenendo

$$x(t+1) = M_1 x(t) \ltimes M_2 x(t) \ltimes \cdots \ltimes M_n x(t)$$

e il membro a destra dell'eguaglianza può essere riscritto tramite il prodotto di una apposita L per il vettore $x(t)$. In definitiva si ha

$$x(t+1) = Lx(t)\ . \tag{2.4}$$

L'equazione appena trovata si dice *forma algebrica* della BN.
La matrice L trovata è una matrice logica di dimensione $2^n \times 2^n$, che è in grado di descrivere totalmente la dinamica della BN attraverso l'equazione (2.4). Si ha infatti il seguente

Teorema 2.1 *La dinamica della BN (2.1) è univocamente determinata dal sistema dinamico lineare (2.4).*

Dimostrazione: A partire dal sistema (2.4) si ricava che $x(t) = L^t x(0)$. L'evoluzione di ogni nodo può essere descritta tramite (2.3), quindi si ha

$$X_i(t+1) = M_i x(t) = M_i L^t x(0)\ ,\ i = 1, \dots, n\ .$$

In questo modo si è dimostrato che si può ricostruire l'andamento di ogni singolo nodo grazie all'equazione algebrica (2.4). $\diamondsuit$

A questo punto si può introdurre un altro tipo di grafo, che rende assai più intuitivo capire il funzionamento della BN.

Definizione 2.3: Il *grafo di stato* è un grafo orientato $\Psi = (\mathcal{V}, \mathcal{A})$. $\mathcal{V}$ è composto da 2^n vertici, $\mathcal{V} = \{x_i = \delta_{2^n}^i, 1 \leq i \leq 2^n\}$, ciascuno associato ad un diverso valore assunto da $x(t)$, cioè alle diverse configurazioni della BN. L'insieme degli archi $\mathcal{A}$ ha cardinalità 2^n, ed è presente un arco nel grafo che va dal vertice i al vertice j se e solo se la colonna i-esima della matrice L è pari a $\delta_{2^n}^j$.

Osservazione 2.2: Ad ogni BN corrisponde un solo grafo di stato, come nel caso del grafo di influenza; per quanto riguarda il viceversa invece, ad ogni grafo di stato corrispondono tutte le reti Booleane che hanno la stessa identica dinamica, e che quindi presentano si funzioni logiche che sono scritte in maniera diversa, ma che poi coincidono a livello di tabella di verità. A meno quindi di differenti modalità di scrittura delle funzioni logiche, ad un dato grafo di stato corrisponde una sola rete Booleana. $\square$

Il grafo di stato appena introdotto è molto utile perchè permette di capire immediatamente come evolve la rete. Se infatti al tempo t lo stato è $x(t) = \delta_{2^n}^i$, $x(t+1)$ sarà pari alla colonna i-esima di L, e questo fatto è facilmente individuabile nel grafo grazie alla presenza di un arco dal nodo x_i al nodo x_j. Avendo quindi una combinazione di valori assunti dalle χ_i al tempo t, si trova il valore di $x(t)$ associato a quella combinazione e osservando il grafico si può subito capire quale sarà la successiva combinazione di χ_i.

Esempio 2.3: Nelle tre matrici successive si riportano le forme algebriche per componente della BN dell'Esempio 2.1, che si possono facilmente ottenere grazie alle tabelle di verità riportate in Tabella 2.1.

$$M_1 = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \quad M_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$M_3 = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

La matrice L presente nella forma algebrica della BN può essere ottenuta con il procedimento spiegato nel capitolo 2, ottenendo:

$$L = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} \delta_8^2 & \delta_8^4 & \delta_8^3 & \delta_8^7 & \delta_8^3 & \delta_8^5 & \delta_8^2 & \delta_8^7 \end{bmatrix} .$$

Dal momento che la lettura della matrice L può risultare pesante quando è scritta nel modo precedente, si può introdurre la seguente notazione

$$L = \delta_8 \begin{bmatrix} 2 & 4 & 3 & 7 & 3 & 5 & 2 & 7 \end{bmatrix} ,$$

che risulta di più facile gestione. Questa notazione verrà spesso usata nel seguito per tutte le matrici logiche.
Il grafo di stato associato alla BN dell'esempio viene riportato in Figura 2.2.

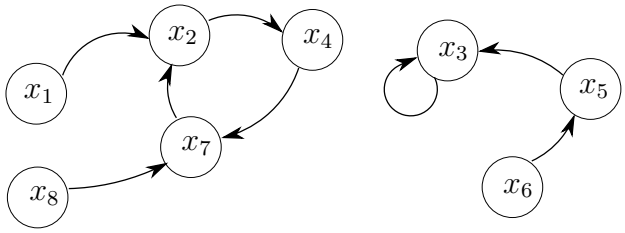


Figura 2.2: Grafo di stato relativo alla BN dell'esempio 3.1.

♣

Questo grafo risulta molto utile per lo studio dei punti fissi e dei cicli del sistema. Nel paragrafo successivo è presente lo studio legato alla ricerca di punti fissi e cicli della rete.

2.3 Punti fissi e cicli

Definizione 2.4: Uno stato x_0 si dice *punto fisso* del sistema (2.4) se $Lx_0 = x_0$.

$\{x_0, Lx_0, \ldots, L^k x_0\}$ si dice *ciclo* del sistema (2.4) di lunghezza k se $L^k x_0 = x_0$ e gli elementi nell'insieme $\{x_0, Lx_0, \ldots, L^{k-1} x_0\}$ sono a due a due distinti.

Osservazione 2.3: Nel grafo di stato un punto fisso è un vertice in cui l'arco uscente rientra nel nodo stesso, quindi un vertice che forma un ciclo di lunghezza 1, mentre un ciclo di lunghezza k nella BN è individuato nel grafo di stato dalla presenza di un ciclo della stessa lunghezza. □

Si riportano nel seguito teoremi e proposizioni presenti in [1] che riguardano questi argomenti; con i primi due teoremi si può ricavare il numero di punti fissi e di cicli della rete.

Teorema 2.2 *Considerando la BN (2.1), $x_0 = \delta^i_{2^n}$ è un suo punto fisso se e solo se nella sua forma algebrica l'elemento $[L]_{ii}$ della matrice L è pari a 1. Questo comporta che il numero di punti fissi della rete, N_e, è pari al numero di i per il quale $[L]_{ii} = 1$, cioè*

$$N_e = \mathrm{tr}(L) \ .$$

Dimostrazione: Considerando $x_0 = \delta^i_{2^n}$, se $[L]_{ii} = 1$, si ha che la colonna i-esima di L è pari a $\delta^i_{2^n}$, per cui $Lx_0 = L\delta^i_{2^n} = \mathrm{Col}_i(L) = \delta^i_{2^n}$, quindi x_0 è un punto fisso. Se viceversa x_0 è un punto fisso si ha $Lx_0 = x_0$ e quindi la colonna i-esima di L deve essere $\delta^i_{2^n}$. ♢

Nel prossimo teorema si utilizza l'insieme dei fattori propri di un numero intero non negativo: dato $k \in \mathbb{Z}_+$, $s \in \mathbb{Z}_+$ si dice *fattore proprio* di k se $s < k$ e $k/s \in \mathbb{Z}_+$. L'insieme dei fattori propri di k verrà denotato con $\mathscr{P}(k)$.

Teorema 2.3 *Nella BN (2.1) il numero di cicli di lunghezza s, denotato con N_s, è induttivamente determinato da*

$$\begin{cases} N_1 = N_e = \mathrm{tr}(L) \\ N_s = \frac{\mathrm{tr}(L^s) - \sum_{k \in \mathscr{P}(s)} k N_k}{s} \ , \ 2 \leq s \leq 2^n \ . \end{cases} \tag{2.5}$$

Dimostrazione: Preso $\delta^i_{2^n}$ elemento appartenente a un ciclo di lunghezza s, si ha ovviamente che $L^s \delta^i_{2^n} = \delta^i_{2^n}$, quindi la colonna i-esima di L^s ha l'elemento diverso da zero in posizione i, il che contribuisce quindi ad aumentare di uno $\mathrm{tr}(L^s)$. Questo succede per tutti gli s elementi di ogni ciclo, per cui $\mathrm{tr}(L^s) \geq sN_s$: i due valori non coincidono perché $\mathrm{tr}(L^s)$

contiene anche gli elementi legati a cicli di lunghezza pari ai fattori propri di s: se infatti $s = rk$, $k \in \mathscr{P}(s)$, e $L^k \delta^i_{2^n} = \delta^i_{2^n}$, si ha $L^s \delta^i_{2^n} = L^{rk} \delta^i_{2^n} = L^k \cdots L^k \delta^i_{2^n} = \delta^i_{2^n}$. Per trovare N_s quindi, alla traccia della matrice L^s vanno sottratti tutti i kN_k con $k \in \mathscr{P}(s)$.
Ovviamente la lunghezza massima dei cicli è 2^n poichè esistono 2^n stati distinti. $\diamondsuit$

Esempio 2.4: Sempre considerando l'Esempio 2.1, si riporta il grafo di stato in cui sono stati evidenziati il punto fisso e il ciclo presenti (Figura 2.3).
Applicando la formula (2.5) si ottiene anche matematicamente che $N_1 =$

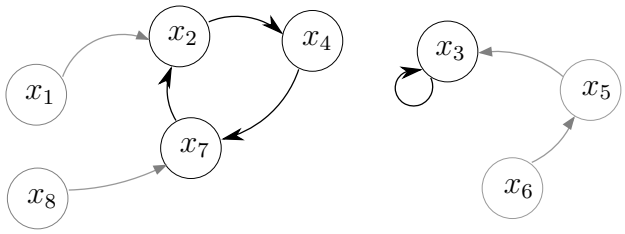


Figura 2.3: Grafo di stato dell'esempio 3.1 con i cicli evidenziati.

1 e $N_3 = 1$, mentre non esistono cicli di lunghezza 2 o maggiore di tre. $\clubsuit$

Definizione 2.5: Le potenze s di L per cui $N_s > 0$ si dicono *potenze non banali*.

Nella prossima proposizione si fornisce un modo per identificare gli stati che appartengono a un determinato ciclo. Si introducono inizialmente due nuovi insiemi che riguardano i primi 2^n interi positivi:

$$C_s = \{i | [L^s]_{ii} = 1\} \ , \ s = 1, 2, \ldots, 2^n, \ s \text{ potenza non banale}$$

$$D_s = C_s \bigcap_{i \in \mathscr{P}(s)} C_i^c \ ,$$

dove $[L^s]_{ii}$ è l'elemento in posizione (i, i) della matrice L^s e C_i^c è il complemento dell'insieme C_i rispetto all'insieme $\{1, 2, \ldots, 2^n\}$.

Proposizione 2.1 *Sia $x_0 = \delta^i_{2^n}$, allora $\{x_0, Lx_0, \ldots, L^s x_0\}$ è un ciclo di lunghezza s se e solo se $i \in D_s$.*

Dimostrazione: Affinché x_0 appartenga a un ciclo di lunghezza s, è necessario che $[L^s]_{ii} = 1$, quindi $x_0 \in C_s$; allo stesso tempo x_0 non deve

appartenere ad un ciclo di lunghezza pari ad un fattore proprio di s, quindi non deve appartenere a C_i , $i \in \mathscr{P}(s)$ e da questo segue la tesi. $\diamondsuit$

Esempio 2.5: Riprendendo l'Esempio 2.1 si ricercano il punto fisso e gli stati appartenenti al ciclo di lunghezza 3. Per quanto riguarda il punto fisso bisogna studiare la matrice L e individuare qual è l'indice i tale che $l_{ii} = 1$. Da una veloce ispezione della matrice L si nota subito che il punto fisso è lo stato $x = \delta_8^3$. Per individuare gli stati appartenenti al ciclo di lunghezza 3 che sappiamo essere presente nella rete, bisogna calcolare il cubo della matrice L. Usando la notazione semplificata introdotta nell'Esempio 2.3, si ha

$$L^3 = \delta_8 \left[\begin{array}{cccccccc} 7 & 2 & 3 & 4 & 3 & 3 & 7 & 4 \end{array} \right] .$$

Con la notazione usata per L^3 è facile convincersi che $[L^3]_{ii} = 1$ coincide col fatto che l'i-esimo elemento riportato in L^3 corrisponde a δ_8^i. Allora si ha

$$C_3 = \{2, 3, 4, 7\} .$$

I fattori propri di 3 sono 1 e 3, e $C_1 = 3$, quindi $C_1^c = \{2, 3, 4, 5, 6, 7, 8\}$. In definitiva si ha che

$$D_s = C_3 \cap C_1^c = \{2, 4, 7\} ,$$

e quindi δ_8^2, δ_8^4, δ_8^7 sono gli stati appartenenti al ciclo di lunghezza 3, fatto facilmente verificabile grazie al grafo in Figura 2.3. $\clubsuit$

Stabiliti i cicli presenti nella rete, è interessante studiare il comportamento dinamico della rete. Di certo esiste un certo istante t dopo il quale lo stato del sistema appartiene a un ciclo, indipendentemente dallo stato iniziale del sistema; questo è dovuto al fatto che esiste un numero finito di stati per cui prima o poi lo stato in cui si arriva appartiene ad un ciclo e in tale ciclo resta in maniera definitiva. Risulta quindi interessante studiare l'istante temporale dopo il quale si può affermare con sicurezza che lo stato è entrato in un ciclo.
Si definisce inizialmente l'insieme Ω, detto insieme *limite*, che consiste di tutti i punti fissi e di tutti gli stati che appartengono a cicli. Ovviamente a regime la BN visiterà solo un sottoinsieme di Ω.
Dal momento che la matrice L è una matrice logica, questa può assumere solo $r := 2^n \times 2^n$ valori distinti; questo comporta che nella sequenza di $r + 1$ elementi

$$L^0 = I_{2^n}, L, L^2, \ldots, L^r$$

devono esserci almeno 2 matrici uguali. Detto r_0 il più piccolo i tale che L^i appare nuovamente nella sequenza, si ha che $r_0 < r$. Da un punto di vista matematico r_0 può essere definito nel seguente modo:

$$r_0 = \underset{0 \le i < r}{\operatorname{argmin}} \left\{ L^i \ \in \ \left\{ L^{i+1}, \ldots, L^r \right\} \right\} \ . \tag{2.6}$$

Proposizione 2.2 *Detto r_0 il valore trovato con (2.6), si ha che, partendo da un qualsiasi stato iniziale, la traiettoria dopo r_0 iterazioni sarà dentro un ciclo.*

Dimostrazione: detto τ il più piccolo numero tale che $L^{r_0} = L^{r_0+\tau}$, se si considera la traiettoria da stato iniziale $x(0) = x_0$ si ha

$$x(1) = Lx_0 \, ; \, x(2) = L^2 x_0 \, ; \, \ldots \, ; \, x(r_0) = L^{r_0} x_0 \, ; \, x(r_0+1) = L^{r_0+1} x_0 \, ; \ldots$$

$$\ldots \, ; \, x(r_0+\tau) = L^{r_0+\tau} x_0 \, ; \, x(r_0+\tau+1) = L^{r_0+\tau+1} x_0 ; \ldots ,$$

quindi si ha $x(r_0) = x(r_0+\tau)$, $x(r_0+1) = x(r_0+\tau+1)$, e così via, quindi la traiettoria è necessariamente dentro un ciclo (che però può avere lunghezza minore di τ). $\diamondsuit$

Definizione 2.6: Dato uno stato iniziale x_0, il *tempo di transizione di* x_0, $T_t(x_0)$, è il più piccolo k tale che $x(0) = x_0$ e $x(k) \in \Omega$.
Il *tempo di transizione della BN*, T_t, è definito nel seguente modo:

$$T_t := \max_{x \in \Delta_{2^n}} \left(T_t(x) \right) .$$

Teorema 2.4 *r_0 definito in (2.6) coincide con il tempo di transizione della BN, cioè*

$$r_0 = T_t \ .$$

Dimostrazione: Riprendendo τ come è stato definito nella dimostrazione della Proposizione 2.2, si ha che $r_0 + \tau \le r$.
Si dimostra inizialmente che se nella rete c'è un ciclo di lunghezza minima ℓ, allora ℓ è un fattore di τ. Se per assurdo non fosse così, allora $\tau = \ell p + s$, $\exists \, p, s, \ 1 \le s < \ell$; se x_0 è uno stato del ciclo di lunghezza ℓ, anche $L^{r_0} x_0$ vi appartiene, ma allora

$$L^{r_0} x_0 = L^{r_0+\tau} x_0 = L^{r_0} L^{\tau} x_0 = L^{r_0} L^s L^{\ell p} x_0 = L^{r_0+s} x_0 \, ;$$

questo risulta assurdo perché $\bar{x} = L^{r_0} x_0$ è uno stato del ciclo di lunghezza ℓ, ma si ha anche che $\bar{x} = L^s \bar{x}$, $s < \ell$, e quindi appartiene anche a un ciclo

di lunghezza s, il che è assurdo perché il ciclo di lunghezza ℓ è minimo. Confrontando la definizione di r_0 e di T_t, si ha che $T_t \leq r_0$ e per assurdo si può supporre che la disuguaglianza sia in senso stretto. Dalla definizione di T_t, per ogni x si ha che $L^{T_t}x$ appartiene ad un ciclo, la lunghezza minima del quale è un fattore di τ, quindi

$$L^{T_t}x = L^{\tau}L^{T_t}x = L^{T_t+\tau}x \ , \ \forall x \ .$$

Quest'ultima uguaglianza comporta che $L^{T_t} = L^{T_t+\tau}$, ma questo è in contraddizione con la definizione di r_0, perché si era supposto che $T_t < r_0$. Si ha quindi che $r_0 = T_t$. ♢

Osservazione 2.4: Visto che per il suo significato T_t risulta sempre minore o uguale di 2^n, si ha quindi che $r_0 \leq 2^n$. □

Osservazione 2.5: τ definito nella dimostrazione della Proposizione 2.2 risulta il minimo comune multiplo della lunghezza di tutti i cicli della BN e viene detto *moltiplicatore di cicli*. □

Esempio 2.6: Considerando il solito Esempio 2.1, si ha

$$L = \delta_8 \left[\begin{array}{cccccccc} 2 & 4 & 3 & 7 & 3 & 5 & 2 & 7 \end{array} \right] \quad L^2 = \delta_8 \left[\begin{array}{cccccccc} 4 & 7 & 3 & 2 & 3 & 3 & 4 & 2 \end{array} \right]$$

$$L^3 = \delta_8 \left[\begin{array}{cccccccc} 7 & 2 & 3 & 4 & 3 & 3 & 7 & 4 \end{array} \right] \quad L^4 = \delta_8 \left[\begin{array}{cccccccc} 2 & 4 & 3 & 7 & 3 & 3 & 2 & 7 \end{array} \right]$$

$$L^5 = \delta_8 \left[\begin{array}{cccccccc} 4 & 7 & 3 & 2 & 3 & 3 & 4 & 2 \end{array} \right] .$$

In definitiva si ha che $L^2 = L^5$, quindi T_t della BN presentata è 2 e il moltiplicatore di cicli τ è pari a 3. Tutto questo comporta che da qualsiasi stato iniziale si parte, dopo 2 iterazioni la rete è entrata in uno dei due cicli, e questo è confermato anche dal grafo di stato della rete (Figura 2.3). Inoltre il valore di τ è una ulteriore conferma che i cicli della rete possono avere lunghezza o 1 o 3, che sono gli unici fattori propri di τ. ♣

2.4 Bacino di attrazione

Altro aspetto interessante dello studio dei cicli è capire a quale ciclo si converge a partire da un determinato stato iniziale. Infatti, visto che la BN è deterministica, ad ogni stato iniziale corrisponde uno e un solo ciclo in cui la rete entra e dove resta indefinitamente.
Se indichiamo con $\mathscr{C}_i$, $i = 1, \ldots, k$, l'insieme che contiene tutti gli stati appartenenti al ciclo i-esimo, si ha ovviamente che $\Omega = \bigcup_{i=1}^{k} \mathscr{C}_i$.

Definizione 2.7: S_i si dice *bacino di attrazione* per $\mathscr{C}_i$ se S_i è l'insieme degli stati che convergono a $\mathscr{C}_i$, cioè $\bar{x} \in S_i$ se e solo se $x(t) = L^t\bar{x} \in \mathscr{C}_i$ per $t > T_t$.

Ovviamente l'unione dei bacini di attrazione S_i crea tutto l'insieme degli stati Δ_{2^n}, e dal momento che gli S_i sono disgiunti, questi formano una partizione di Δ_{2^n}.
Osservazione 2.6: Nel grafo di stato Ψ di una BN, i diversi bacini di attrazione possono essere individuati tramite la seguente procedura:

- per ogni i, j, se nel grafo diretto è presente l'arco (i, j) dal vertice x_i al vertice x_j, si aggiunge all'insieme $\tilde{\mathcal{A}}$ dei lati del nuovo grafo $\tilde{\Psi}$ l'elemento (j, i). Il grafo $\tilde{\Psi}$ che ne risulta è indiretto.

- si trovano le diverse componenti connesse del grafo $\tilde{\Psi}$; ogni componente contiene uno e un solo ciclo, e i vertici che la compongono formano il bacino di attrazione del ciclo stesso.

A questo punto risulta immediato stabilire anche il tempo di transizione della rete tramite il grafo di stato; basta infatti considerare per ogni componente connessa gli eventuali stati che non appartengono al ciclo e trovare la lunghezza del percorso tra lo stato considerato e il primo stato appartenente al ciclo. Per ogni bacino si memorizza la lunghezza maggiore e poi si trova tra tutte queste la maggiore in assoluto e in questo modo si ottiene il tempo di transizione della rete. □

Per trovare i diversi bacini di attrazione risulta utile il concetto di *stato antecedente (diretto)*: $q \in \Delta_{2^n}$ è antecedente di $p \in \Delta_{2^n}$ se $p = Lq$. Con $L^{-1}(p)$ si indica l'insieme di tutti gli stati antecedenti di p.

Proposizione 2.3 *Dato $\mathscr{C}_i$, ciclo della BN, il suo bacino di attrazione è composto dall'insieme*

$$S_i = \mathscr{C}_i \cup L^{-1}(\mathscr{C}_i) \cup L^{-2}(\mathscr{C}_i) \cup \ldots \cup L^{-T_t}(\mathscr{C}_i) \ .$$

Dimostrazione: Il bacino di attrazione di $\mathscr{C}_i$ contiene ovviamente tutti gli stati appartenenti al ciclo, ma anche tutti quelli che in una iterazione arrivano nel ciclo (cioè $L^{-1}(\mathscr{C}_i)$) e così via fino a quelli che in T_t iterazioni entrano nel ciclo; ovviamente non ci saranno altri stati in grado di entrare nel ciclo con un numero maggiore di iterazioni, proprio per definizione di T_t. ♢

Per calcolare gli stati antecedenti di uno stato p si può sfruttare la seguente

Proposizione 2.4 *Gli stati antecedenti (un passo o k passi) dello stato p si trovano nel seguente modo:*

$$\begin{cases} L^{-1}(p) = \{\delta_{2^n}^j | Col_j(L) = p\} \\ L^{-k}(p) = \{\delta_{2^n}^j | Col_j(L^k) = p \quad k = 2, \ldots, T_t\} \end{cases}$$

Esempio 2.7: Nella BN dell'Esempio 2.1 ci sono 2 cicli, uno composto da un solo stato, $\mathscr{C}_1 = \{3\}$, e uno composto da 3 stati, $\mathscr{C}_2 = \{2, 4, 7\}$. Per il bacino relativo a $\mathscr{C}_1$ si ha

$$S_1 = \{3\} \cup \{3, 5\} \cup \{3, 5, 6\} = \{3, 5, 6\} \ ,$$

mentre per il bacino relativo a $\mathscr{C}_2$ si ha

$$S_2 = \{2, 4, 7\} \cup \{1, 2, 4, 7, 8\} \cup \{1, 2, 4, 7, 8\} = \{1, 2, 4, 7, 8\} \ .$$

Anche nella ricerca del bacino di attrazione dei diversi cicli il grafo di stato fornisce tutte le informazioni necessarie (Figura 2.3). ♣

2.5 Stabilità

In questo paragrafo si cercano le condizioni per cui una rete converga ad un unico stato, cioè perché sia globalmente stabile.

Definizione 2.8: Una BN si dice *globalmente convergente* se il suo insieme limite Ω consiste di un unico punto fisso. La convergenza globale si dice anche *stabilità globale*.

Nel caso in cui la rete presenta un solo punto fisso, e quindi un solo ciclo di lunghezza uno, tutti gli stati convergono al punto fisso perché il suo bacino di attrazione è obbligatoriamente tutto Δ_{2^n} (dal momento che la suddivisione in bacini è una partizione dell'insieme di tutti gli stati).

Osservazione 2.7: La stabilità globale di una BN si riflette nel suo grafo di stato con la presenza di un solo ciclo di lunghezza 1. Inoltre il grafo $\tilde{\Psi}$, ricavato nell'Osservazione 2.6 aggiungendo i lati invertiti, deve ovviamente avere una sola componente connessa. □

Per stabilire la convergenza globale di una BN vale la seguente

Proposizione 2.5 *La BN (2.1), con la relativa forma algebrica (2.4), è globalmente convergente se e solo se una delle seguenti condizioni equivalenti è soddisfatta:*

1. *l'unica potenza non banale di L è 1 e* $\mathrm{tr}(L) = 1$*;*

2. *il moltiplicatore di cicli è 1 e* $\mathrm{tr}(L^{2^n}) = 1$*;*

3. *le colonne di L^{T_t} sono tutte uguali.*

Dimostrazione: Per il primo punto, se la BN è globalmente stabile allora presenta un solo ciclo, e questo è di lunghezza uno, quindi si ha che $\mathrm{tr}(L) = 1$ e l'unica potenza non banale è 1; viceversa, se $\mathrm{tr}(L)$=1 la rete presenta un solo ciclo di lunghezza uno, e poiché l'unica potenza non banale è 1, l'insieme limite della BN è composto da un solo punto fisso, e quindi la rete è globalmente stabile.
Per quanto riguarda il secondo punto, se la BN è globalmente stabile, allora esiste un unico ciclo, e questo è di lunghezza 1; il moltiplicatore di cicli τ deve quindi essere pari a 1, essendo il minimo comune multiplo delle lunghezze di tutti i cicli, e anche $\mathrm{tr}(L^{2^n})$ deve essere pari a uno, altrimenti si avrebbe più di un punto fisso. Viceversa, il fatto che τ sia pari a uno implica che la rete ha solo punti fissi, e grazie al fatto che $\mathrm{tr}(L^{2^n}) = 1$, la rete presenta un solo punto fisso, quindi la rete è globalmente convergente per definizione.
Per il terzo punto, se la rete è globalmente convergente allo stato $\delta^i_{2^n}$, esaurito il transitorio si ha che lo stato raggiunto dalla rete è il punto fisso, quindi L^{T_t} ha tutte le colonne pari a $\delta^i_{2^n}$. Viceversa, se tutte le colonne di L^{T_t} sono uguali a $\delta^i_{2^n}$, vuol dire che dopo T_t iterazioni la rete ha raggiunto lo stato $\delta^i_{2^n}$ a partire da qualsiasi stato iniziale, e quindi la rete è globalmente stabile. ♢

Osservazione 2.8: Nel caso della terza condizione, ovviamente si ha che dalla potenza T_t di L in poi la matrice risulta avere tutte le colonne uguali, quindi la conoscenza del periodo di transizione non è fondamentale; basta infatti controllare se L^{2^n} ha tutte le colonne uguali. In caso contrario si può affermare che la rete non è globalmente convergente (si ha infatti che $T_t \leq 2^n$). □

Esempio 2.8: L'Esempio 1,usato dall'inizio del capitolo, non rappresenta ovviamente una rete globalmente stabile, dal momento che presenta, oltre al punto fisso anche un ciclo di lunghezza 3. Da un punto di vista del grafo (Figura 2.3) si nota subito che con il procedimento descritto nell'Osservazione 2.6 si ottengono 2 componenti connesse distinte, una formata dai vertici $\{1, 2, 4, 7, 8\}$ e l'altra dai rimanenti vertici.
Prendendo in considerazione un'altra BN, descritta dalla seguente forma

algebrica

$$x(t+1) = Lx(t) = \delta_8 \left[\begin{array}{cccccccc} 1 & 1 & 1 & 3 & 4 & 5 & 2 & 6 \end{array} \right] x(t) \, ,$$

con L scritta in maniera semplificata, questa invece è globalmente convergente. Si verifica facilmente infatti che l'unica potenza non banale di L è 1 e che la traccia della matrice L è pari a 1, oppure si vede subito che elevando L all'ottava potenza, si ottiene come risultato una matrice formata da tutte colonne pari a δ_8^1. Ovviamente il punto fisso a cui converge la rete è lo stato δ_8^1.
Studiando il grafo di stato della rete in esame, riportato in Figura 2.4, si nota subito che esiste una sola componente connessa e che il periodo di transitorio è pari a 5. ♣

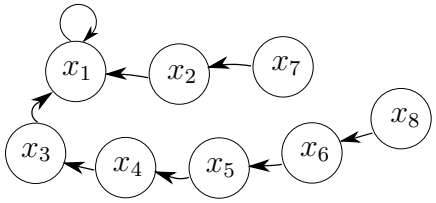


Figura 2.4: Grafo di rete relativo alla seconda rete dell'esempio 3.8.

Nell'ultimo paragrafo del capitolo si applicano le nozioni relative alle matrici positive e alle catene di Markov riportate nel paragrafo 1.5, per portare la matrice L nella sua forma normale.

2.6 Forma normale della matrice L

La BN descritta in forma algebrica presenta un legame molto forte con le catene di Markov; in effetti risulta immediato dimostrare che il sistema algebrico che descrive una BN presenta le stesse proprietà del sistema algebrico che descrive l'evoluzione di una catena di Markov appena si considera la sua versione trasposta. Inoltre per le sue caratteristiche intrinseche è una catena di Markov molto facile perché deterministica.
L'evoluzione di una catena di Markov è descritta dall'equazione riportata in (1.8), dove $x(t)$ è un vettore stocastico, e la matrice P è stocastica per righe. Se si effettua la trasposizione dell'equazione si ottiene

$$x(t+1) = P^T x(t)$$

dove ovviamente P^T è una matrice stocastica per colonne.
Analizzando il sistema algebrico che descrive una BN, il vettore $x(t)$ che

dello stato è un vettore canonico, quindi stocastico, e anche le colonne della matrice L sono vettori canonici, per cui L risulta stocastica per colonne. Il sistema algebrico che rappresenta la BN descrive quindi una catena di Markov, in cui tutto però risulta deterministico. Lo stato iniziale infatti è noto, e da ogni stato si passa ad uno e un solo stato, per cui non sono necessarie delle considerazioni probabilistiche come nel caso più generale delle catene di Markov.

Osservazione 2.9: Una BN vista come catena di Markov presenta 2^n stati $S_1, \ldots, S_n$; visto che il vettore $x(t)$ è canonico, e visto il suo significato nel contesto delle catene di Markov, questo significa che se $x(t) = \delta^i_{2^n}$, allora al tempo t la catena di Markov è con probabilità 1 nello stato i. Esiste quindi un nesso biunivoco tra il valore dello stato $x(t) = \delta^i_{2^n}$ del sistema algebrico di una BN e lo stato S_i della catena di Markov ad esso associata. Nel seguito, quando si parla di stato i si intende lo stato S_i della catena di Markov associata alla BN, che è indissolubilmente legato al vettore $x = \delta^i_{2^n}$. □

La relazione di comunicazione tra stati definita nella Definizione 1.12 si riflette nella seguente partizione degli stessi nel caso in esame:

- ogni insieme di stati che forma un ciclo della rete dà vita a una classe ergodica;
- gli stati che non appartengono a cicli formano classi di comunicazione transitorie composte da un solo elemento.

Osservazione 2.10: In base alla Proposizione 1.5 si può affermare con certezza che ogni BN presenta sempre almeno 1 classe ergodica e quindi un ciclo. □

Utilizzando la numerazione degli stati già presentata per le catene di Markov, si ottiene per permutazione una matrice $\tilde{L}$, che fornisce informazioni importanti sulla dinamica del sistema. Il livello più profondo (usando per livello lo stesso significato introdotto per le catene di Markov) che si trova distinguendo gli stati di una BN in livelli corrisponde al valore assunto da T_t. All'istante T_t infatti a partire da qualsiasi stato iniziale la rete è entrata in un ciclo, ed esiste almeno uno stato $\bar{x}$ che entra in un ciclo dopo esattamente T_t passi, e quindi il livello più profondo corrisponde a questo valore.
Se la BN in esame presenta $k > 1$ cicli, con l'etichettatura degli stati appena presentata si ottiene la seguente matrice $\tilde{L}$ in forma normale:

$$\tilde{L} = \left[\begin{array}{ccc|cccc} \tilde{L}_1 & \mathbf{0} & \mathbf{0} & \star & \cdots & \cdots & \star \\ \mathbf{0} & \ddots & \mathbf{0} & \star & \cdots & \cdots & \star \\ \mathbf{0} & \mathbf{0} & \tilde{L}_k & \star & \cdots & \cdots & \star \\ \hline & & & 0 & \star & \cdots & \star \\ & & & & 0 & \vdots & \star \\ & & & & & \ddots & \star \\ & & & & & & 0 \end{array}\right] .$$

Essa è caratterizzata da k blocchi diagonali formati da matrici irriducibili, ognuno dei quali rappresenta un ciclo della rete, mentre i restanti blocchi diagonali sono nulli di dimensione 1 e le relative colonne presentano l'elemento unitario sopra la diagonale.
Come è già stato notato nel caso delle catene di Markov, la forma assunta dalla matrice $\tilde{L}$ corrisponde alla forma normale della matrice L di partenza. Questa può essere ottenuta attraverso trasformazione di cogredienza ed è sempre possibile farlo grazie al Teorema 1.4.

Osservazione 2.11: Se si ricava la forma normale della matrice L numerando gli stati con lo stesso procedimento esposto per le catene di Markov, la matrice $\tilde{L}$ che si ottiene ha in realtà una forma più caratterizzata: le colonne relative agli stati di primo livello seguono le colonne relative agli stati dei cicli (che sono le prime colonne), e presentano l'elemento 1 nelle righe relative alle classi dei cicli; le colonne relative agli stati di secondo livello seguono quelle relative agli stati del primo, e presentano l'uno in corrispondenza delle righe delle classi del primo livello e così via per le restanti classi.
Se invece si calcola la forma normale solamente cercando una matrice di permutazione che porti L in forma normale, questo non è assicurato; si può solo affermare con sicurezza che considerati due stati i e j che non appartengono a cicli, se partendo dallo stato i al passo successivo si arriva in j, allora obbligatoriamente lo stato $\tilde{i}$ che corrisponde nella numerazione precedente allo stato i, e lo stato $\tilde{j}$ che corrisponde allo stato j, sono tali per cui $\tilde{j} < \tilde{i}$. □

Osservazione 2.12: Per ottenere la forma normale di L si opera una trasformazione di cogredienza, cioè si considerano gli stati in un ordine diverso da quello iniziale. Per ricostruire lo stato del sistema di partenza conoscendo lo stato del sistema dopo la trasformazione, si può usare

$$x = \Pi\tilde{x} \ , \tag{2.7}$$

con Π la matrice di permutazione che porta gli stati nell'ordine desiderato. Esiste quindi una diversa associazione tra il valore del nuovo stato $\tilde{x}$ e il valore delle variabili logiche χ_i che compaiono in (2.1). $\square$

Portando quindi in forma normale la matrice L si possono ottenere numerose informazioni sulla struttura della BN, in particolare su:

- numero di cicli presenti e loro lunghezza;
- suddivisione degli stati in bacini di attrazione e tempo di transitorio.

Per quanto riguarda i cicli, i blocchi isolati $\tilde{L}_i$ indicano i cicli della rete, e la dimensione di ciascun blocco indica la lunghezza del ciclo legato ad esso. Ovviamente conoscendo la lunghezza di ogni ciclo si può ottenere il moltiplicatore di cicli τ come minimo comune multiplo delle diverse lunghezze.
Per ricostruire i bacini di attrazione è invece sufficiente elevare a potenza la matrice $\tilde{L}$ più volte, fino a che l'elemento pari a uno di ogni colonna non appartenga alle righe delle classi relative ai cicli, cioè fino a che

$$\tilde{L}^r = \left[\begin{array}{ccc|ccc} \tilde{L}_1^r & & & S_{1,h+1}^r & \cdots & S_{1,2^n}^r \\ & \ddots & & \vdots & \vdots & \vdots \\ & & \tilde{L}_1^r & S_{k,h+1}^r & \cdots & S_{k,2^n}^r \\ \hline & \mathbf{0} & & & \mathbf{0} & \end{array}\right] \tag{2.8}$$

con h pari alla somma delle dimensioni dei blocchi $\tilde{L}_i,\ i = 1, \ldots, k$ e $S_{j,\ell}^r,\ j = 1, \ldots, k,\ \ell = h+1, \ldots, 2^n$, vettore colonna di dimensione pari a quella di $\tilde{L}_j$ e tale che su ogni colonna di $\tilde{L}^r$ con indice maggiore di h un solo vettore è diverso dal vettore nullo ed è un vettore canonico.
Stabilire a quale bacino di attrazione lo stato $\tilde{x} = \delta_{2^n}^i, i = h+1, \ldots, 2^n$, appartenga risulta immediato: basta infatti trovare il vettore $S_{j,i}^r$ che risulta diverso dal vettore nullo, e l'indice j stabilisce a quale ciclo giunge il sistema a partire dallo stato iniziale $\delta_{2^n}^i$. Ovviamente per avere la stessa informazione relativa a x, cioè con la base di partenza, basta trovare la corrispondenza tra gli stati nelle due diverse basi.
Il valore minimo di r per cui la matrice $\tilde{L}^r$ assume la forma desiderata corrisponde ovviamente al tempo di transitorio della rete, cioè $T_t = r$.

Esempio 2.9: Riprendendo in esame la BN dell'Esempio 2.1, si può trovarne la forma normale. Analizzando il grafo di stato è facile convincersi

che operando la permutazione degli stati indicata da

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 3 & 2 & 4 & 7 & 1 & 8 & 5 & 6 \end{pmatrix}$$

l'ordine in cui compaiono gli stati è quello indicato per ottenere la matrice in forma normale. Operando infatti la trasformazione di cogredienza utilizzando la matrice Π_σ associata alla permutazione σ si ha:

$$\tilde{L} = \Pi_\sigma^T L \Pi_\sigma = \left[\begin{array}{c|ccc|ccc|c} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] .$$

La suddivisione della matrice è stata fatta per evidenziare la considerazione espressa nell'Osservazione 2.11. Utilizzando questa nuova matrice l'evoluzione della BN si esprime come $\tilde{x}(t+1) = \tilde{L}\tilde{x}(t)$, con $\tilde{x}$ il vettore di stato espresso nella nuova base.
La corrispondenza tra lo stato $\tilde{x}$ nella nuova base e le variabili logiche $\chi_1, \ldots, \chi_n$ si ottiene trovando prima il corrispondente valore dello stato nella base precedente, applicando la formula (2.7), e poi applicando la formula (1.1). La relazione che si trova effettuando i calcoli è riassunta nella tabella 2.2.

$\tilde{x}$	x	χ_1	χ_2	χ_3
δ_8^1	δ_8^3	1	0	1
δ_8^2	δ_8^2	1	1	0
δ_8^3	δ_8^4	1	0	0
δ_8^4	δ_8^7	0	0	1
δ_8^5	δ_8^1	1	1	1
δ_8^6	δ_8^8	0	0	0
δ_8^7	δ_8^5	0	1	1
δ_8^8	δ_8^6	0	1	0

Tabella 2.2: Conversione nuovo stato $\tilde{x}$, vecchio stato x e variabili logiche χ_1, χ_2, χ_3.

Con la matrice in forma normale risulta subito evidente che esistono 2 cicli, uno di lunghezza 1 relativo allo stato 1 (con il nuovo riordino), e

uno di lunghezza 3 relativo agli stati 2,3,4.
Per quanto riguarda i bacini di attrazione, elevando alla seconda potenza la matrice $\tilde{L}$ si ottiene la forma (2.8), della quale nel seguito si riporta solo il blocco T contenente i diversi $S^2_{i,j}$, perché risulta la parte interessante per stabilire i bacini di attrazione:

$$T = \left[\begin{array}{c|c|c|c} 0 & 0 & 1 & 1 \\ \hline 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right] .$$

Analizzando T si conclude che gli stati 5 e 6 appartengono al bacino di attrazione del ciclo di lunghezza 3, mentre gli stati 7 e 8 al bacino del punto fisso. Dal momento che la potenza 2 è quella minima per avere $\tilde{L}^k$ nella forma (2.8), il tempo di transitorio corrisponde a 2 istanti.
Guardando la relazione tra lo stato nella nuova e nella vecchia base, si nota che dall'analisi di $\tilde{L}$ e delle sue potenze sono stati ottenuti gli stessi risultati già ottenuti con le altre tecniche presentate. ♣

3 Reti Booleane di controllo

Una naturale estensione delle reti Booleane è l'inserimento nelle diverse equazioni di ulteriori variabili logiche che sono controllabili dall'esterno, cioè l'inserimento di ingressi di controllo. In questo contesto quindi si analizzano anche i concetti di raggiungibilità, controllabilità, osservabilità e ricostruibilità. Non tutti questi concetti verranno trattati in maniera approfondita, dal momento che solo alcuni sono fondamentali per lo scopo della tesi. Trattandosi comunque di proprietà importanti per comprendere il modo di operare di una rete Booleana di controllo, vengono riportate almeno le considerazioni più significative.

3.1 Definizione, forma algebrica e grafo di una rete Booleana di controllo

Definizione 3.1: Una *rete Booleana di controllo* è un sistema di equazioni logiche del tipo

$$\begin{cases} \chi_1(t+1) = f_1(\chi_1(t), \ldots, \chi_n(t), \upsilon_1(t), \ldots, \upsilon_m(t)) \\ \qquad\qquad \vdots \\ \chi_n(t+1) = f_n(\chi_1(t), \ldots, \chi_n(t), \upsilon_1(t), \ldots, \upsilon_m(t)) \end{cases} \tag{3.1}$$

e di equazioni logiche

$$\gamma_j(t) = h_j(\chi_1(t), \ldots, \chi_n(t)),\ j = 1, \ldots, p. \tag{3.2}$$

Le variabili $\chi_i \in \mathcal{B},\ i = 1, \ldots, n$, sono i nodi della BCN, le variabili $\upsilon_\ell \in \mathcal{B},\ \ell = 1, \ldots, m$, sono gli ingressi di controllo e le variabili $\gamma_j \in \mathcal{B},\ j = 1, \ldots, p$, sono le uscite.
Le funzioni $f_i : \mathcal{B}^{n+m} \to \mathcal{B},\ i = 1, \ldots, n$, e $h_j : \mathcal{B}^n \to \mathcal{B},\ j = 1, \ldots, p$, sono funzioni logiche.

Con l'abbreviazione BCN (Boolean Control Network) si indica nel seguito una rete Booleana di controllo.
Come per le BN, anche in questo caso è possibile trovare la forma algebrica del sistema.
A questo scopo si introducono le variabili vettoriali

$$X_i \in \Delta_2,\ U_\ell \in \Delta_2,\ Y_j \in \Delta_2$$

che rappresentano le rispettive variabili logiche $\chi_i \in \mathcal{B},\ \upsilon_\ell \in \mathcal{B},\ \gamma_j \in \mathcal{B},\ i = 1, \ldots, n,\ \ell = 1, \ldots, m,\ j = 1, \ldots, p$, con il sistema già visto nei

capitoli precedenti.
Attraverso l'uso delle variabili

$$x(t) = \ltimes_{i=1}^{n} X_i(t) \in \Delta_{2^n},\ u(t) = \ltimes_{l=1}^{m} U_\ell(t) \in \Delta_{2^m},\ y(t) = \ltimes_{j=1}^{p} Y_j(t) \in \Delta_{2^p},$$

si può costruire la seguente forma algebrica della BCN:

$$\begin{cases} x(t+1) = L \ltimes u(t) \ltimes x(t) \\ y(t) = H \ltimes x(t) \end{cases}, \tag{3.3}$$

con $L \in \mathcal{L}^{2^n \times 2^{n+m}}$ e $H \in \mathcal{L}^{2^p \times 2^n}$.
Per trovare l'equazione relativa a $y(t)$ basta applicare il procedimento spiegato nella sezione 1.4, mentre per ritrovare quella relativa a $x(t+1)$ il procedimento risulta leggermente più complesso per la presenza dell'ingresso $u(t)$. A seconda del diverso valore assunto da $u(t)$, il rapporto (e quindi la matrice) che lega $x(t+1)$ a $x(t)$ è diverso. Ad ogni diverso ingresso $u(t) = \delta_{2^m}^i$ è quindi associata una diversa matrice L_i, ed facile verificare che utilizzando il prodotto semitensoriale $L \ltimes u(t) = L \ltimes \delta_{2^m}^i$ il risultato è una matrice quadrata logica di dimensione 2^n che corrisponde alla sottomatrice di L relativa alle colonne di indice $(i-1)2^n+1, \ldots, i2^n$. La matrice L è quindi costruita accostando le diverse $L_i,\ i = 1, \ldots, 2^m$, calcolate utilizzando come valore di ingresso $u(t) = \delta_{2^m}^i$:

$$L = \begin{bmatrix} L_1 & L_2 & \ldots & L_{2^m} \end{bmatrix}.$$

Esempio 3.1: Data la seguente BCN

$$\begin{cases} \chi_1(t+1) = (\chi_1(t) \wedge \chi_2(t)) \vee \upsilon_1(t) \\ \chi_2(t+1) = \chi_2(t) \wedge \upsilon_2(t) \end{cases}$$

$$\gamma(t) = \chi_1(t) \leftrightarrow \chi_2(t)$$

si può, dopo alcuni calcoli, ottenere la sua forma algebrica che presenta le seguenti matrici:

$$L = \left[\begin{array}{cccc|cccc|cccc|cccc} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \end{array}\right]$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}.$$

La scelta di usare una rete Booleana molto ridotta per l'esempio è giustificata dalle dimensioni della matrice L che si ottiene, che già in questo caso risulta alquanto onerosa da scrivere. ♣

Come nel caso delle reti Booleane, anche per le reti di controllo Booleane è possibile introdurre un grafo di stato. L'insieme dei vertici del grafo è $\mathcal{V} = \{x_i = \delta_{2^n}^i, 1 \leq i \leq 2^n\}$ come nel caso delle BN, mentre per trovare l'insieme degli archi uscenti dal nodo x_i, si calcola per ogni ingresso $u = \delta_{2^m}^j,\ j = 1, \ldots, 2^m$ la matrice $L \ltimes u$, si considera la sua i-esima colonna, che sarà pari a $\delta_{2^n}^k,\ 1 \leq k \leq 2^n$, e si inserisce in $\mathcal{A}$, l'insieme contenente gli archi del grafo, l'arco orientato (x_i, x_k).

Esempio 3.2: Il grafo di stato della rete di controllo Booleana presentata nell'Esempio 3.1 viene riportato in Figura 3.1.

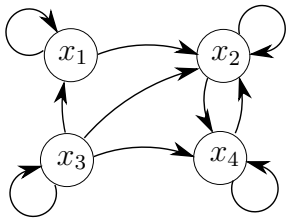


Figura 3.1: Grafo di rete relativo alla BCN dell'Esempio 3.1.

♣

Il grafo di stato permette di capire immediatamente verso quali stati può spostarsi il sistema a partire da uno stato dato. Nel caso si volesse mantenere l'informazione relativa a quali ingressi creano un determinato arco tra 2 vertici, basterebbe etichettare ogni arco con il valore degli ingressi che lo genera, ma renderebbe pesante la sua rappresentazione grafica.

3.2 Interpretazione come switched system

Una rete di controllo Booleana può anche essere interpretata come una rete Booleana che può cambiare ad ogni istante la sua matrice di stato. Come è già stato notato infatti il prodotto $L \ltimes u(t)$, con $u(t) = \delta_{2^m}^i$ è pari ad una matrice quadrata L_i di dimensione 2^n che per l'istante t funge da matrice di stato di una BN che in quel momento è equivalente alla BCN

di partenza.
Una rete di controllo Booleana può quindi essere descritta come un Boolean switched system del tipo

$$x(t+1) = L_{\sigma(t)}x(t)$$

dove $\sigma(t)$ è una sequenza di numeri presi dall'insieme $\{1, \dots, 2^m\}$ e ogni $L_i,\ i = 1, \dots, 2^m$ viene detto sottosistema della rete.

Esempio 3.3: Riprendendo l'Esempio 3.1, la BCN in esame può essere interpretata come uno switched system che ha la possibilità di commutare tra 4 diverse BN e le 4 diverse matrici che caratterizzano le diverse reti sono:

$$L_1 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad L_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$L_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \qquad L_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}.$$

A seconda dell'ingresso che viene fornito alla rete quindi la dinamica del sistema sarà dettata dalla rispettiva matrice L_i. ♣

3.3 La matrice L_{tot}

Una matrice che risulta molto utile, soprattuto per lo studio della raggiungibilità del sistema, è la seguente

$$L_{tot} = \bigvee_{i=1}^{2^m} L_i\,;$$

in particolare $[L_{tot}]_{ij} = 1$ se e solo se esiste almeno un L_i che abbia in posizione (i, j) l'elemento 1.
Con questa matrice si capisce verso quali stati si può andare a partire da un determinato stato, ma si perde l'informazione relativa a quale ingresso permetta effettivamente di spostarsi nello stato desiderato. Se infatti lo stato al tempo t è $x(t) = \delta_{2^n}^i$, con la colonna i-esima di L_{tot} si possono ottenere tutti gli stati raggiungibili da $x(t)$, che sono quelli in corrispondenza agli elementi non nulli del vettore, ma evidentemente è impossibile

stabilire quale ingresso sia necessario per andare in uno di quegli stati. Con la matrice L_{tot} risulta anche più facile la costruzione del grafo di stato di una BCN: per stabilire gli archi uscenti dal vertice x_i è sufficiente guardare la colonna i-esima di L_{tot} e, per ogni suo elemento pari a uno, mettere un arco verso il rispettivo vertice (se $[\mathrm{Col}_i(L_{tot})]_j = 1$, allora è presente un arco tra x_i e x_j).

Esempio 3.4: La matrice L_{tot} relativa all'Esempio 3.1 è la seguente:

$$L_{tot} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} .$$

♣

Osservazione 3.1: Ciò che è importante nella matrice L_{tot} è la sparsità della matrice stessa. Nel seguito sono utilizzate le potenze di questa matrice quadrata, ma, poiché risulta importante solo la sparsità e non i valori numerici assunti dagli elementi diversi da zero, con L_{tot}^{τ} si intende in realtà la matrice Booleana associata a L_{tot}^{τ}; viene comunque usata la stessa notazione per non appesantire ulteriormente la scrittura. □

3.4 Comportamento asintotico e forma normale di L_{tot}

Per una BCN le considerazioni che si possono fare sul comportamento asintotico della rete sono molto meno significative rispetto a quelle che si possono fare nel caso di reti Booleane senza controllo. Anche gli stati di una rete controllata possono essere suddivisi in classi di comunicazione, usando lo stesso criterio delle reti Booleane, cioè 2 stati x_i e x_j appartengono alla stessa classe se e solo se esiste un percorso nel grafo di stato che permette di passare da x_i a x_j e viceversa. Questa condizione, come verrà dimostrato successivamente, si riflette sulla matrice L_{tot} nel fatto che devono esistere τ_i e τ_j tali per cui $[L_{tot}^{\tau_i}]_{ji} = 1$ e $[L_{tot}^{\tau_j}]_{ij} = 1$.
Per le reti in esame le classi di comunicazione possono essere suddivise nel seguente modo:

- classi *chiuse*, cioè quelle in cui non si può passare da uno stato di quella classe a uno stato di un'altra, qualunque sia la sequenza di ingressi utilizzata;

- classi *transitorie*, cioè quelle in cui esistono stati dai quali si può passare in stati di altre classi attraverso opportuni ingressi. A differenza delle reti Booleane, le classi transitorie di una BCN possono essere composte anche da più di un elemento.

Osservazione 3.2: Le classi chiuse non sono state definite ergodiche perchè nel caso di reti controllate non è detto che si converga necessariamente a una di queste; la presenza infatti di classi transitorie composte da più di un elemento implica l'esistenza di cicli anche in classi non chiuse. Questo comporta che se ad un certo istante la rete è in uno stato appartenente a una classe transitoria composta da più di un elemento, applicando una opportuna sequenza di ingressi si può restare in quella classe indefinitivamente, senza raggiungere una classe chiusa. □

Osservazione 3.3: Anche per le BCN si può affermare che esiste sempre almeno una classe di comunicazione chiusa, e questo è dovuto al numero di stati possibili del sistema, che è finito. □

Grazie alla suddivisione in classi di comunicazione è possibile fare le seguenti considerazioni sulla dinamica del sistema:

- se lo stato iniziale del sistema appartiene a una classe chiusa, la dinamica evolverà solo all'interno di questa classe, qualunque sia la sequenza di ingressi usata;

- solo per le classi transitorie che sono formate da un unico stato e per il quale non esiste un ingresso che porti lo stato in se stesso è possibile affermare che asintoticamente la rete non sarà in quelle classi;

- per tutte le le altre classi transitorie è possibile trovare una sequenza di ingressi che mantiene lo stato sempre in quella classe. Ciononostante, nel caso all'istante t lo stato esca dalla classe, si ha la certezza che non potrà più tornarci in futuro;

- a differenza delle reti Booleane non è detto che ad una classe transitoria corrisponda una sola classe chiusa, perché dipende tutto dalla sequenza di ingressi usata.

Se si porta attraverso una trasformazione di cogredienza la matrice L_{tot} nella sua forma normale (trasposta), si può anche in questo caso ottenere tutte le informazioni relative alle classi di comunicazione: ai blocchi isolati corrisponderanno le classi chiuse, e agli altri blocchi diagonali le

classi transitorie; ovviamente i blocchi diagonali di dimensione 1 che presentano l'elemento 0 corrispondono agli stati in cui asintoticamente il sistema non potrà mai essere.

Esempio 3.5: In Figura 3.2 si riporta il grafo di stato in cui sono state evidenziate le 3 classi di comunicazione che sono presenti nella rete dell'Esempio 3.1. La classe di comunicazione chiusa è quella formata dagli stati x_2 e x_4. Nel seguito si riporta la forma normale della matrice L_{tot} e

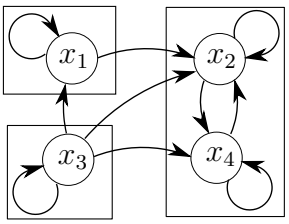


Figura 3.2: Classi di comunicazione nel grafo di rete dell'esempio 3.1.

la permutazione usata per ottenerla:

$$\begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \sigma = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 2 & 1 & 3 \end{pmatrix} .$$

Con questo esempio risulta evidente che non si può fare nessuna affermazione certa sul comportamento asintotico del sistema. ♣

3.5 Individuazione dei cicli

Per lo studio delle strategie di controllo delle BCN risulta utile la conoscenza dei cicli presenti nelle reti Booleane di controllo. È stata quindi creata una funzione ricorsiva per individuare tutti i cicli (semplici) di una BCN. Nel seguito si riporta l'algoritmo in maniera descrittiva e in pseudocodice (Algoritmo 3.1).
All'algoritmo ricorsivo vengono forniti in ordine crescente tutti gli stati della BCN, ed esso trova tutti i cicli che partono dallo stato dato, x_p, e che non coinvolgono gli stati x_i, $1 \leq i < p$ (perchè sicuramente sono già stati individuati quando l'algoritmo è stato precedentemente chiamato con x_i come stato di partenza per i cicli).
L'algoritmo memorizza una successione di stati (che comincia con x_p), e

la relativa successione di ingressi che permettono quell'evoluzione dello stato; quando viene chiamato, l'algoritmo analizza l'ultimo stato della successione, e in particolare identifica tutti gli stati in cui la BCN può portarsi a partire dall'ultimo. A questo punto ci sono 4 possibili situazioni riguardo ad ognuno degli stati in cui si giunge:

- lo stato non appartiene alla successione degli stati e non coincide con nessun x_i, $1 \leq i < p$, per cui viene aggiunto a quest'ultima, si aggiorna la successione degli ingressi con l'ingresso che permette di arrivare in questo stato e si richiama l'algoritmo;

- lo stato appartiene alla successione degli stati, ma non coincide con x_p; in questo caso non si richiama l'algoritmo perchè, anche nel caso in cui si trovasse una sequenza di ingressi per cui lo stato torna in x_p, il ciclo individuato non sarebbe semplice, e quindi non risulta interessante;

- lo stato in cui si arriva è pari a x_i, $1 \leq i < p$, per cui l'eventuale ciclo che coinvolge x_p e x_i è già stato individuato quando l'algoritmo è ha individuato i cicli con x_i come stato iniziale;

- lo stato corrisponde ad x_p, per cui la successione di stati e di ingressi (a cui si deve aggiungere l'ultimo usato per arrivare in x_p) identifica un ciclo della BCN.

L'algoritmo ricorsivo è ovviamente destinato a terminare dopo al massimo 2^n chiamate successive, dal momento che quando uno stato si ripete, l'algoritmo non effettua un'ulteriore chiamata a se stesso, e gli stati della rete sono 2^n.
L'implementazione in Matlab di questa funzione può essere trovata in Appendice A.

Per descrivere un ciclo $\mathscr{C}$ di periodo T di una BCN, si utilizza la seguente notazione:

$$\mathscr{C} = (x(t), u(t)), (x(t+1), u(t+1)), \ldots, (x(t+T-1), u(t+T-1)) \ ,$$

con $L \ltimes u(t+T-1) \ltimes x(t+T-1) = x(t+T) = x(t)$.

Osservazione 3.4: L'algoritmo implementato risulta effettivamente utile solo quando la rete presenta un numero di stati e di ingressi relativamente piccolo, poiché il tempo di esecuzione cresce molto velocemente all'aumentare di questi fattori (come succede nel caso dell'applicazione

Algoritmo 3.1 Algoritmo "**IndividuaCicli**" per l'individuazione dei cicli di una BCN

Input: z, contenente la successione degli stati già visitati; ing, contenente la successione degli ingressi; L, matrice di transizione della BCN; x_p stato di inizio (e quindi di fine) del ciclo;

Output: un vettore contenente tutti i cicli trovati che partono dallo stato coincidente con il primo elemento di z;

---

su=ultimo stato della successione;
sa=vettore contenente gli stati in cui si può arrivare da su;
for elementi di sa coincidenti con x_p **do**
    si aggiorna il vettore contenente i cicli;
end for
for elementi di sa non presenti in z e maggiori di x_p **do**
    sr=stato di sa in analisi;
    ir=ingresso da fornire per raggiungere sr da su;
    $z = [z\ sr]$;
    $ing = [ing\ ir]$;
    IndividuaCicli(z, ing, L, x_p);
end for

---

pratica trattata nel capitolo 5). □

Esempio 3.6: Applicando l'algoritmo appena esposto alla BCN dell'Esempio 3.1, si determina che i cicli presenti nella rete in analisi sono i seguenti:

$$\mathscr{C}_1 = (1,1);\ \mathscr{C}_2 = (1,3);\ C_3 = (2,1);\ C_4 = (2,2);$$

$$\mathscr{C}_5 = ((2,3),(4,1))\,;\ \mathscr{C}_6 = ((2,3),(4,2))\,;\ \mathscr{C}_7 = ((2,4),(4,1))\,;$$
$$\mathscr{C}_8 = ((2,4),(4,2))\,;\ \mathscr{C}_9 = (3,3);\ C_{10} = (4,3);\ \mathscr{C}_{11} = (4,4)\,.$$

♣

3.6 Raggiungibilità e controllabilità

I concetti che sono presentati in questo paragrafo riguardano la possibilità di guidare lo stato del sistema verso un preciso valore intervenendo ovviamente sull'ingresso.

Definizione 3.2: Lo stato $x_f = \delta_{2^n}^i$ è raggiungibile da $x_0 = \delta_{2^n}^j$ in τ passi se esiste una sequenza di ingressi $u(0), u(1), \ldots, u(\tau-1)$ tale che partendo da $x(0) = x_0$ il sistema si porta allo stato x_f all'istante $t = \tau$.

Teorema 3.1 *$x_f = \delta_{2^n}^i$ è raggiungibile da $x_0 = \delta_{2^n}^j$ in τ passi se e solo se $[L_{tot}^\tau]_{ij} = 1$.*

Dimostrazione: Entrambe le implicazioni vengono dimostrate tramite induzione.
Supponendo che x_f sia raggiungibile in 1 passo da x_0, esiste $u(0) = \delta_{2^m}^k$ tale che $L \ltimes u(0) \ltimes x_0 = L_k x_0 = x_f$, quindi si ha che $[L_k]_{ij} = 1$ e di conseguenza $[L_{tot}]_{ij} = 1$.
Per ipotesi induttiva si suppone che se x_f è raggiungibile in $\tau - 1$ passi da x_0, allora $[L_{tot}^{\tau-1}]_{ij} = 1$.
Si consideri quindi x_f raggiungibile da x_0 in τ passi; questo comporta l'esistenza di una sequenza di ingressi $\bar{u}_0, \ldots, \bar{u}(\tau-2), \bar{u}(\tau-1)$ che porta lo stato da x_0 a x_f. Partendo da x_0 e utilizzando l'ingresso $\bar{u}(0), \ldots, \bar{u}(\tau-2)$ si giunge ad uno stato $x_d = \delta_{2^n}^h$, quindi per ipotesi induttiva $[L_{tot}^{\tau-1}]_{hj} = 1$, ma si ha anche che $[L_{tot}]_{ih} = 1$. In definitiva si ha

$$[L_{tot}^\tau]_{ij} = \sum_{k=1}^{2^n} [L_{tot}]_{ik} [L_{tot}^{\tau-1}]_{kj} = 1 \tag{3.4}$$

perchè le matrici in considerazione sono positive, e per lo meno per l'indice h i due fattori sono positivi.
L'implicazione inversa si dimostra in maniera simile: se $[L_{tot}]_{ij} = 1$, per definizione di L_{tot} esiste un ingresso $\bar{u}(0)$ tale che $L \ltimes \bar{u}(0)$ ha come j-esima colonna il vettore canonico $\delta_{2^n}^i$, per cui $L \ltimes \bar{u}(0) \ltimes \delta_{2^n}^j = \delta_{2^n}^i$, quindi $x_f = \delta_{2^n}^i$ è raggiungibile in un passo da $x_0 = \delta_{2^n}^j$.
Per ipotesi induttiva si suppone che se $[L_{tot}^{\tau-1}]_{ij} = 1$ allora $x_f = \delta_{2^n}^i$ è raggiungibile in $\tau - 1$ passi da $x_0 = \delta_{2^n}^j$. Considerando allora $[L_{tot}^\tau]_{ij} = 1$, questa può essere scritta come in (3.4), e quindi deve esistere k per il quale $[L_{tot}^{\tau-1}]_{kj} = 1$ e $[L_{tot}]_{ik} = 1$. La prima condizione comporta che $x_d = \delta_{2^n}^k$ è raggiungibile in $\tau - 1$ passi da $x_0 = \delta_{2^n}^i$, mentre la seconda comporta che $x_f = \delta_{2^n}^i$ è raggiungibile da x_d in 1 passo, quindi x_f è raggiungibile da x_0 in τ passi. ♢

Osservazione 3.5: Nel grafo di stato, la raggiungibilità di x_f da x_0 in τ passi implica l'esistenza di un percorso di lunghezza τ tra il nodo x_0 e il nodo x_f. □

L'insieme di tutti gli stati raggiungibili da x_0 in k passi viene indicato con $R_k(x_0)$.

Definizione 3.3: x_f si dice *raggiungibile* da x_0 se esiste τ e un ingresso $u(0), \ldots, u(\tau-1)$ grazie al quale l'evoluzione del sistema parte da x_0 e arriva in x_f.

Il Teorema che segue è in realtà una generalizzazione di quello precedente:

Teorema 3.2 *$x_f = \delta_{2^n}^i$ è raggiungibile da $x_0 = \delta_{2^n}^j$ se e solo se* $\exists\, \tau > 0 \,|\, [L_{tot}^\tau]_{ij} = 1$.

Visto che gli stati sono in numero finito e pari a 2^n, se lo stato x_f non è raggiungibile da x_0 in $1, \ldots, 2^n$ passi, allora non può esserlo con nessun altro numero di passi, quindi $R(x_0)$, insieme degli stati raggiungibili da x_0 si trova con la seguente formula

$$R(x_0) = \bigcup_{i=1}^{2^n} R_i(x_0)\ .$$

Introducendo allora la matrice

$$\mathbb{L} = \bigvee_{i=1}^{2^n} L_{tot}^i$$

si hanno tutte le informazioni necessarie per stabilire la raggiungibilità tra i vari stati, dal momento che se $[\mathbb{L}]_{ij} = 1$ allora lo stato $x_f = \delta_{2^n}^i$ è raggiungibile dallo stato $x_0 = \delta_{2^n}^j$; ovviamente con questa matrice non è possibile trovare il numero di passi con cui x_f è raggiungibile da x_0.
Osservazione 3.6: La suddivisione degli stati in classi di comunicazione è legata alla possibilità di uno stato $x_i = \delta_{2^n}^i$ di comunicare con un altro stato $x_j = \delta_{2^n}^j$ e il viceversa. Questo ovviamente comporta che x_j sia raggiungibile da x_i e che x_i sia raggiungibile da x_j. □

Definizione 3.4: Una BCN si dice *raggiungibile da* x_0 se tutti gli stati del sistema sono raggiungibili a partire da esso, quindi se $R(x_0) = \Delta_{2^n}$.

Definizione 3.5: Una BCN si dice *controllabile* a x_f se quest'ultimo è raggiungibile da ogni altro stato .

Definizione 3.6: Una BCN si dice *globalmente controllabile*, o semplicemente *controllabile*, se essa è raggiungibile da ogni $x_0 \in \Delta_{2^n}$.

È facile provare il seguente

Teorema 3.3 *Una BCN è controllabile se e solo se la sua matrice L_{tot} è irriducibile.*

Osservazione 3.7: Una condizione equivalente al Teorema precedente è che la matrice $\mathbb{L}$ sia strettamente positiva. □

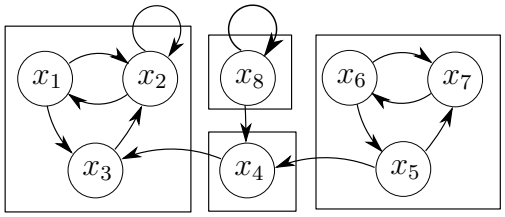
Osservazione 3.8: Alla luce della Sezione 3.4, quest'ultimo teorema implica che un sistema controllabile ha una sola classe di comunicazione che contiene tutti gli stati (e quindi è chiusa).
La presenza di uno stato x_f a cui il sistema è controllabile implica invece che il sistema presenti una sola classe chiusa, a cui x_f appartiene, e automaticamente implica che il sistema sia controllabile a tutti gli altri eventuali stati che appartengono a quella classe. □

Esempio 3.7: Si consideri una rete Booleana di controllo che ha come matrice della sua forma algebrica la seguente

$$L = \delta_8 \left[\begin{array}{cccccccc|cccccccc} 2 & 1 & 2 & 3 & 4 & 7 & 6 & 8 & 3 & 2 & 2 & 3 & 7 & 5 & 6 & 4 \end{array} \right] ,$$

scritta nella sua forma semplificata. La matrice L_{tot} e il suo grafo di stato sono presenti successivamente:

$$L_{tot} = \left[\begin{array}{cccc|cccc} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \right]$$



Con la numerazione degli stati adottata, la matrice L_{tot} risulta già in forma normale, ed è quindi facile vedere che esiste una sola classe chiusa e altre 3 classi transitorie.
Se per esempio si vogliono trovare tutti gli stati raggiungibili dallo stato

x_7 si analizza la settima colonna di L_{tot} e delle sue prime 8 potenze:

$$\mathrm{Col}_7(L_{tot}) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad \mathrm{Col}_7(L_{tot}^2) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad \mathrm{Col}_7(L_{tot}^3) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad \mathrm{Col}_7(L_{tot}^4) = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

$$\mathrm{Col}_7(L_{tot}^5) = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad \mathrm{Col}_7(L_{tot}^6) = \mathrm{Col}_7(L_{tot}^7) = \mathrm{Col}_7(L_{tot}^8) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Analizzando le colonne si nota che l'unico stato non raggiungibile in un numero finito di passi è lo stato x_8, mentre tutti gli altri sono raggiungibili con un numero di passi minore di 7. Con questo esempio si nota anche che se $x_i \in R_k(x_0)$ non è detto che $x_i \in R_{k+1}(x_0)$, come succede per lo stato x_6 che è raggiungibile in un passo ma non in 2.
Riportando la matrice $\mathbb{L}$ si possono fare delle considerazioni molto generali:

$$\mathbb{L} = \left[\begin{array}{ccc|c|ccc|c} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ \hline 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

La matrice è stata suddivisa secondo le classi di comunicazione: come è stato già sottolineato a livello teorico, l'unica classe chiusa è raggiungibile da tutti gli stati del sistema, e quindi il sistema stesso è controllabile a tutti gli stati di questa classe, quindi a x_1, x_2, x_3, e non esistono altri stati a cui è controllabile. La classe transitoria di dimensione 3 ha tutti gli stati che sono raggiungibili tra di loro, come è logico aspettarsi dal momento che per avere dimensione 3, la classe deve necessariamente presentare

almeno un ciclo tra i suoi stati. Ultima cosa da notare è che lo stato x_8 è raggiungibile solo da se stesso, come risulta evidente dal grafo di stato della rete. ♣

Dopo aver trovato un modo per studiare la raggiungibilità tra i diversi stati analizzando la matrice $\mathbb{L}$, resta da trovare un modo in cui scegliere la sequenza di ingresso per passare da uno stato all'altro. Supponendo allora che lo stato $x_f = \delta_{2^n}^j$ sia raggiungibile da $x_0 = \delta_{2^n}^i$, si vuole trovare una sequenza che permetta di passare da x_0 a x_f. Ovviamente non è detto che esista un unico ingresso che permette questo passaggio, e non è nemmeno detto che le sequenze di ingresso che lo permettono abbiano tutte la stessa lunghezza. Nel seguito si espone un metodo per trovare una sequenza di ingresso di lunghezza minima τ^* grazie alla quale sia possibile passare da $x(0) = x_0$ a $x(\tau^*) = x_f$. Il valore di τ^* corrisponde al numero minimo di passi per cui la raggiungibilità è possibile, cioè

$$\tau^* = \min_{\tau \in \{1,\dots,2^n\}} \{\tau \mid [L_{tot}^\tau]_{ji} = 1\} .$$

Se x_f è raggiungibile in $\tau^* > 1$ passi da x_0, necessariamente esiste un $x_d = \delta_{2^n}^k$ raggiungibile da x_0 in $\tau^* - 1$ passi e tale che x_f è raggiungibile da x_d in un passo, dal momento che τ^* è la lunghezza del percorso minimo tra i due stati. Si può allora trovare un ingresso che permette di passare da x_d a x_f, e il valore trovato corrisponde all'ultimo valore da dare all'ingresso, quindi a $u(\tau^* - 1)$. Si ripete lo stesso ragionamento tra x_0 e x_d che sono raggiungibili in $\tau^* - 1$ passi trovando $u(\tau^* - 2)$, e continuando a reiterare il ragionamento finché non si trova uno stato raggiungibile da x_0 in un passo, caso in cui basta trovare l'ingresso che permetta questo passaggio, e la sequenza di ingressi è stata trovata.
Nel seguito (Algoritmo 3.2) si riporta l'algoritmo in maniera più schematica.

Con questo algoritmo è quindi possibile ricostruire una (minima) sequenza di ingressi che permette di passare da x_0 a x_f.

3.7 Stabilizzabilità

In questo paragrafo si studia la possibilità di condurre la rete Booleana ad un determinato stato e di mantenervela per tutta la storia futura del sistema.

Definizione 3.7: Una BCN è stabilizzabile allo stato $x_e = \delta_{2^n}^i$ se per ogni $x_0 \in \Delta_{2^n}$ esiste una sequenza di ingressi $u(t)$ e un istante $\tau \geq 0$ per cui si ha $x(t) = x_e$ per ogni $t \geq \tau$.

Algoritmo 3.2 Algoritmo per la ricostruzione della sequenza di ingressi per portare lo stato da x_0 a x_f.

Input: L_{tot}, $L = [L_1|\dots|L_{2^m}]$, $x_0 = \delta_{2^n}^i$, $x_f = \delta_{2^n}^j$, τ^*;
Output: $u(0), \dots, u(\tau^* - 1)$;

while $\tau^* > 1$ **do**
  $c_i = \mathrm{Col}_i(L_{tot}^{\tau^*-1})$;
  $r_j = \mathrm{Riga}_j(L_{tot})$;
  $x = c_i \wedge r_j^T$;
  k=indice primo elemento diverso da 0 in x;
  $s = 0$;
  while $\mathrm{Col}_k(L_s) \neq x_f$ **do**
    $s++$;
  end while
  $u(\tau^* - 1) = \delta_{2^m}^s$;
  $x_f = \delta_{2^n}^k$;
end while
$s = 0$;
while $\mathrm{Col}_i(L_s) \neq x_f$ **do**
  $s++$;
end while
$u(0) = \delta_{2^m}^s$;

Ovviamente affinché si possa stabilizzare il sistema allo stato $x_e = \delta_{2^n}^i$, deve esistere un ingresso $u = \delta_{2^m}^j$ tale per cui $[L_j]_{ii} = 1$, cioè x_e è un punto fisso di una delle sottoreti della BCN. Inoltre x_e deve essere raggiungibile da ogni stato del sistema, e quindi il sistema deve essere controllabile a x_e.

Teorema 3.4 *Una rete di controllo Booleana è stabilizzabile allo stato $x_e = \delta_{2^n}^i$ se e solo se valgono le due seguenti condizioni:*

1. *x_e è un punto fisso del j-esimo sottosistema della rete, per qualche $j \in \{1, 2, \ldots, 2^m\}$;*
2. *x_e è raggiungibile da ogni stato iniziale $x(0)$, cioè*

$$x_e \in \bigcap_{x(0) \in \Delta_{2^n}} R(x(0)) \, .$$

Dimostrazione: Se la rete è stabilizzabile allo stato x_e, vuol dire che esiste un ingresso in grado di portare un qualsiasi stato iniziale allo stato finale x_e, quindi quest'ultimo è raggiungibile da ogni altro stato della rete; inoltre per definizione per tutti gli istanti $t \geq \tau$ il sistema resta sempre nello stato x_e, quindi deve esistere un ingresso $u = \delta_{2^m}^k$ che porta lo stato x_e in se stesso, e quindi è un punto fisso per il k-esimo sottosistema della BCN.
Viceversa, il secondo punto assicura che esiste τ per cui, partendo da un qualsiasi $x(0)$, $x(\tau)$ risulta pari a x_e, e il primo punto assicura che esiste un ingresso grazie al quale $x(t) = x_e$ per ogni $t \geq \tau$. ♢

Come conseguenza immediata di questo ultimo Teorema e di alcune considerazioni del paragrafo precedente si può enunciare il seguente

Corollario 3.1 *La BCN è stabilizzabile allo stato $x_e = \delta_{2^n}^i$ se e solo se*

1. *$[L_{tot}]_{ii} = 1$;*
2. *la riga i-esima di $\mathbb{L}$ è pari a $\mathbf{1}_{2^n}$.*

Osservazione 3.9: Anche dal grafo di stato è possibile stabilire se una BCN è stabilizzabile a un determinato stato x_e; è sufficiente infatti controllare che esista un self-loop per lo stato x_e e che esista un percorso in grado di portare ogni altro stato del sistema fino a x_e. □
Osservazione 3.10: Per le considerazioni fatte nel paragrafo precedente, affinché sia possibile stabilizzare il sistema a x_e, il sistema deve presentare una sola classe di comunicazione chiusa e x_e deve obbligatoriamente

appartenerci (ed essere un punto fisso per un qualche ingresso). Questo è conseguenza del fatto che da una classe chiusa non si può mai uscire, per cui o x_e appartiene all'unica classe chiusa del sistema per cui è possibile la stabilizzazione, o gli stati che appartengono ad una classe chiusa a cui x_e non appartiene non potranno mai raggiungere lo stato desiderato. Se una BCN presenta più di una classe chiusa, non esisterà quindi alcuno stato a cui il sistema sia stabilizzabile. □

Corollario 3.2 *Se il sistema è globalmente controllabile, allora è stabilizzabile a tutti gli stati per cui* $[L_{tot}]_{ii} = 1$

Esempio 3.8: Riprendendo in esame l'Esempio 3.7, gli unici 2 stati che presentano $[L_{tot}]_{ii} = 1$ sono x_2 e x_8, ma solo la seconda riga di $\mathbb{L}$ è formata da tutti 1, per cui l'unico stato a cui quella rete è stabilizzabile è x_2. ♣

Per trovare la sequenza di ingressi per stabilizzare il sistema si può usare l'Algoritmo 3.2, prendendo x_e come stato finale e come stato iniziale quello in cui la rete è nel momento in cui si vuole cominciare il processo di stabilizzazione.

3.8 Osservabilità e ricostruibilità

In questo paragrafo vengono trattati i concetti di osservabilità e di ricostruibilità per una BCN; i concetti non verranno trattati in maniera esaustiva, perchè presentano un'importanza non essenziale per quanto riguarda le strategie di controllo che vengono presentate successivamente. Ciononostante essi costituiscono un argomento importante, che va a completare lo studio di queste reti. Gran parte dei risultati qui riportati provengono da [3].

Definizione 3.8: Una BCN si dice *osservabile* se per ogni coppia di stati inziali $x_1 = \delta_{2^n}^i$ e $x_2 = \delta_{2^n}^j$, $i \neq j$, e una qualsiasi sequenza di ingresso $u(t)$, $t \in \mathbb{Z}_+$ l'evoluzione dell'uscita $y_1(t)$ (con stato inziale x_1) e $y_2(t)$, $t \in \mathbb{Z}_+$ (con stato iniziale x_2), sono distinte.

Osservazione 3.11: Esiste una definizione alternativa per l'osservabilità, quella proposta in [1], per la quale il sistema è osservabile se per ogni coppia di stati iniziali esiste un ingresso grazie al quale le evoluzioni delle due corrispondenti uscite sono distinte. Quest'ultima definizione è molto

meno stringente di quella usata in [3], nel senso che ogni sistema osservabile secondo la Definizione 3.8 è osservabile anche secondo la definizione proposta in [1]. □

Un sistema osservabile permette di determinare lo stato iniziale della BCN conoscendo i valori dell'ingresso e dell'uscita della rete. Si possono dare delle condizioni necessarie e sufficienti per stabilire se una rete è osservabile, e per enunciarle si introduce il concetto di *traiettorie stato-ingresso*. Una traiettoria stato-ingresso è un sequenza di coppie $(x(t), u(t))$, $t \in \mathbb{Z}_+$ che il sistema descrive quando sollecitato dall'ingresso $u(\cdot)$. Una traiettoria stato-ingresso si dice *periodica di periodo* $k > 0$ se $(x(t), u(t)) = (x(t+k), u(t+k))$, $\forall t \in \mathbb{Z}_+$. Nel caso in cui nella traiettoria periodica non esista $h < k \mid (x(t), u(t)) = (x(t+h), u(t+h))$, la traiettoria si dice periodica di periodo minimo k. In maniera abbreviata si può descrivere una traiettoria periodica come k coppie ordinate

$$((x_1, u_1), (x_2, u_2), \ldots, (x_k, u_k))$$

dove

$$\begin{cases} x_{\ell+1} = L \ltimes u_\ell \ltimes x_\ell, \ \forall \ell \in \{1, \ldots, k-1\} \\ x_1 = L \ltimes u_k \ltimes x_k \end{cases}$$

Il teorema successivo si riporta senza dimostrazione formale:

Teorema 3.5 *Una BCN (3.3) è osservabile se e solo se*

1. *per ogni* $x_1, x_2 \in \Delta_{2^n}$ *e per ogni* $u \in \Delta_{2^m}$, *la condizione* $(x_1, u) \neq (x_2, u)$ *e* $L \ltimes u \ltimes x_1 = L \ltimes u \ltimes x_2$ *implica* $Hx_1 \neq Hx_2$*;*

2. *per ogni coppia di traiettorie stato-ingresso periodiche dello stesso periodo minimo* k*, descritte da due* k*-uple distinte*

   $$((x_1, u_1), \ldots, (x_k, u_k)) \neq ((\bar{x}_1, u_1), \ldots, (\bar{x}_k, u_k))$$

   le corrispondenti traiettorie di uscita sono periodiche di periodo k *e descritte da due* k*-uple distinte*

   $$(Hx_1, Hx_2, \ldots, Hx_k) \neq (H\bar{x}_1, H\bar{x}_2, \ldots, H\bar{x}_k)\,.$$

Osservazione 3.12: La prima condizione viene detta *distinguibilità degli stati prima della loro fusione*: essa implica che, dati due stati distinti per i quali esiste uno stesso ingresso che porta in un passo la BCN al medesimo stato, questi due stati devono creare due distinte uscite;

la seconda condizione viene detta *distinguibilità degli stati appartenenti a cicli*, e implica che due distinte traiettorie stato-ingresso periodiche dello stesso periodo minimo k, guidate dallo stesso ingresso $u(1), \ldots, u(k)$, devono corrispondere a due distinte traiettorie di uscita periodiche di periodo k.
Ovviamente queste due condizioni sono necessarie per avere l'osservabilità, mentre per la dimostrazione della sufficienza si rimanda a [3] □

Osservazione 3.13: Si può dimostrare che se la BCN è osservabile, la conoscenza di $u(t)$ e $y(t)$ per $t \in [0, N^2]$ è sufficiente per determinare univocamente $x(0)$. In realtà si può ulteriormente restringere l'arco di tempo per la determinazione di $x(0)$: si dimostra infatti che è sufficiente la conoscenza di ingresso e uscita per $t \in [0, T+1]$, con

$$T = \frac{(N+1)(N-2)}{2}$$

per poter determinare con certezza quale era lo stato iniziale della BCN. □

Osservazione 3.14: Introducendo la *matrice di osservabilità* in h passi relativa alla sequenza di ingressi $u(0) = \delta_{2^m}^{i_0}, \ldots, u(h-2) = \delta_{2^m}^{i_{h-2}}$

$$\mathcal{O}_{u,h} = \begin{bmatrix} H \\ HL_{i_0} \\ HL_{i_1}L_{i_0} \\ \vdots \\ HL_{i_{h-2}} \cdots L_{i_0} \end{bmatrix},$$

è facile convincersi che la i-esima colonna di questa matrice fornisce la successione di uscite $y(0), \ldots, y(h-1)$ (incolonnate una sopra l'altra) che si ottengono con la BCN con stato iniziale $x(0) = \delta_{2^n}^i$ e l'ingresso riportato in precedenza.
Con la definizione di osservabilità usata in [3] e dall'Osservazione precedente, una BCN è osservabile se e solo se per qualsiasi scelta della sequenza di ingressi la matrice $\mathcal{O}_{u,N^*}$, con $N^* = T + 1 = N(N-1)/2$, presenta tutte le colonne distinte.
Considerando la definizione data in [1] invece, una BCN è osservabile se e solo se impilando le diverse $\mathcal{O}_{u,N^*}$ ottenibili con tutte le possibili sequenze di ingresso $u(t)$, tutte le colonne sono distinte. □

Utilizzando il concetto espresso nella prossima Definizione, si può trovare una soluzione alternativa per uno dei problemi di controllo con retroazione dall'uscita che verrà esposto nel capitolo successivo:

Definizione 3.9: Una BCN si dice *ricostruibile* se esiste $T \in \mathbb{Z}_+$ tale che, per ogni sequenza di ingressi, la conoscenza simultanea di traiettorie di ingresso e uscita per gli istanti $t \in [0, T]$ permette di determinare unicamente $x(T)$.

Per la dimostrazione del Teorema successivo si rimanda sempre a [3]:

Teorema 3.6 *Una BCN è ricostruibile se e solo se per ogni coppia di traiettorie stato-uscita periodiche dello stesso periodo minimo k, descritte dalle due k-uple ordinate*

$$((x_1, u_1), \ldots, (x_k, u_k)) \neq ((\bar{x}_1, u_1), \ldots, (\bar{x}_k, u_k)) \ ,$$

le corrispondenti traiettorie di uscita sono periodiche di periodo k e descritte da due differenti k-uple

$$(Hx_1, Hx_2, \ldots, Hx_k) \neq (H\bar{x}_1, H\bar{x}_2, \ldots, H\bar{x}_k) \ .$$

Come è facile notare, la condizione che deve verificarsi coincide con la seconda condizione che compare nel Teorema 3.5, quindi si ha che l'osservabilità di una BCN implica la sua ricostruibilità, mentre il viceversa non è vero.
Osservazione 3.15: Ovviamente, per quanto detto nell'Osservazione 3.13, se la BCN è ricostruibile, allora lo stato finale può essere ricostruito entro N^* passi. □

Esempio 3.9: Considerando nuovamente l'Esempio 3.7 e analizzando la sua matrice L e il grafo di stato, si ricava che per soddisfare la condizione di distinguibilità degli stati prima della loro fusione la matrice H deve soddisfare i seguenti vincoli:

$$H\delta^1_{2^n} \neq H\delta^3_{2^n} \qquad H\delta^1_{2^n} \neq H\delta^4_{2^n} \qquad H\delta^2_{2^n} \neq H\delta^3_{2^n} \ .$$

Per la condizione relativa alla distinguibilità di stati appartenenti a cicli, le traiettorie stato-ingresso periodiche che possono creare problemi sono le seguenti:

$$((1,1),(2,1)) \quad ((6,1),(7,1))$$
$$((3,2),(2,1),(1,2)) \quad ((5,2),(7,1),(6,2)) \quad ((5,2),(7,2),(6,2)) \ ,$$

dalle quali si ricavano i seguenti vincoli:

$$H\delta^1_{2^n} \neq H\delta^2_{2^n} \qquad H\delta^6_{2^n} \neq H\delta^7_{2^n} \qquad (H\delta^1_{2^n}, H\delta^2_{2^n}) \neq (H\delta^6_{2^n}, H\delta^7_{2^n})$$

$$(H\delta_{2^n}^2, H\delta_{2^n}^1) \neq (H\delta_{2^n}^6, H\delta_{2^n}^7) \qquad (H\delta_{2^n}^1, H\delta_{2^n}^3, H\delta_{2^n}^2) \neq (H\delta_{2^n}^6, H\delta_{2^n}^5, H\delta_{2^n}^7)$$
$$(H\delta_{2^n}^5, H\delta_{2^n}^7, H\delta_{2^n}^6) \neq (H\delta_{2^n}^7, H\delta_{2^n}^6, H\delta_{2^n}^5)\,.$$

Si può allora avere, a seconda della matrice stato-uscita, una BCN osservabile, e quindi ricostruibile, una BCN solo ricostruibile, o una BCN né osservabile né ricostruibile. Per poter soddisfare tutte le richieste, sono necessarie almeno 3 uscite distinte per quanto riguarda la y della forma algebrica, e quindi la BCN deve necessariamente presentare almeno 2 variabili logiche di uscita per avere l'osservabilità o la ricostruibilità della rete. Per questo la matrice stato-uscita deve avere un numero di righe pari almeno a 4. Per esempio con la seguente matrice H_1, la BCN risulta osservabile:

$$H_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix},$$

mentre con la seguente H_2 la BCN è solo ricostruibile, dal momento che non viene osservato il vincolo $H\delta_{2^n}^1 \neq H\delta_{2^n}^4$:

$$H_1 = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

Infine con la matrice H_3 non si rispetta né lo stesso vincolo precedente, né il primo vincolo imposto sui cicli, e quindi la BCN non è nemmeno ricostruibile:

$$H_3 = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

♣

Con quest'ultimo argomento si conclude il capitolo relativo alle reti di controllo Booleane; nel prossimo vengono riportate delle strategie di controllo per queste reti: retroazione dallo stato, retroazione dall'uscita e controllo ottimo.

4 Strategie di Controllo per BCN

In questo capitolo vengono trattati diversi tipi di controllo che possono essere utilizzati nel caso di BCN; ovviamente si suppone sempre di poter decidere, senza nessun vincolo esterno, quale sia il valore dell'ingresso da dare in un determinato istante.
Per i primi due tipi di controllo, che vengono illustrati nei paragrafi 4.1 e 4.2, l'obiettivo sarà quello di condurre la rete verso un determinato stato e di mantenervela indefinitivamente. In altri termini si vuole trovare un ingresso in grado di stabilizzare la BCN ad uno stato, essendo a conoscenza nel primo caso dello stato in cui la rete è in ogni istante, mentre nell'altro caso la conoscenza è limitata all'uscita della BCN in ogni istante. Nel caso del controllo ottimo, presentato nel paragrafo 4.3, l'obiettivo invece è quello di trovare un ingresso in modo tale da minimizzare o massimizzare un indice di comportamento, e questo comporterà, come si vedrà in seguito, di guidare la rete, attraverso il controllo, verso un ciclo e di mantenervela all'infinito.

4.1 Retroazione dallo stato

Lo scopo della retroazione dallo stato è quello di generare un ingresso istantaneo, dipendente dal valore dello stato della rete nell'istante attuale, che sia in grado di portare la rete in uno stato $x_e \in \Delta_{2^n}$ e di mantenervela per gli istanti successivi. Il tipo di retroazione che si applica in questo contesto è una retroazione statica e tempo variante: ad ogni stato corrisponde un determinato valore di ingresso per tutto il tempo in cui si controlla la rete. Un modo per descrivere la retroazione di stato è tramite l'uso di una matrice $K \in \mathcal{L}^{2^m \times 2^n}$ con la quale si può calcolare l'ingresso $u(t)$ nel seguente modo:

$$u(t) = Kx(t) \, .$$

Ovviamente visto che la retroazione è tempo invariante, la matrice K è costante nel tempo.
Se lo scopo è quello di stabilizzare il sistema allo stato x_e, è necessario che il sistema sia controllabile a questo stato, e deve esistere almeno un ingresso $u = \delta_{2^m}^j$ grazie al quale, per il sottosistema associato L_j, lo stato x_e è un punto fisso; in altre parole la rete deve essere stabilizzabile allo stato x_e.
Con la prossima Proposizione si va a dimostrare che, se il sistema è stabilizzabile ad uno stato x_e, è possibile farlo attraverso una retroazione;

l'enunciato e la dimostrazione sono un caso particolare di una Proposizione riportata in [4], in cui l'enunciato riguarda in generale cicli e non soltanto punti fissi.

Proposizione 4.1 *Se una BCN (3.3) è stabilizzabile ad un qualche punto fisso $x_e = \delta_{2^n}^i$, allora è stabilizzabile mediante una retroazione dallo stato.*

Dimostrazione: Visto che la BCN è stabilizzabile, devono valere le condizioni riportate nel Teorema 3.4; si vuole dimostrare che grazie a queste condizioni si può ricavare la matrice K di retroazione. Dalla prima condizione si ricava che esiste $u = \delta_{2^m}^j$ tale che

$$x_e := \delta_{2^n}^i = L \ltimes \delta_{2^m}^j \ltimes \delta_{2^n}^i = L \ltimes u \ltimes x_e \, .$$

Quindi, se $\mathrm{Col}_i(K) = \delta_{2^m}^j$, si ha che, quando $x = x_e$, l'ingresso di retroazione $u = Kx$ vale $u = \delta_{2^m}^j$, ed è proprio un ingresso adatto all'obiettivo di mantenere fisso il punto x_e.
Si consideri ora una partizione di Δ_{2^n} in insiemi $\mathcal{S}_i$, $i \in Z_+$, tali che $\mathcal{S}_i$ contiene gli stati da cui x_e è raggiungile in i passi, ma non in meno di i (con $\mathcal{S}_0$ si indica l'insieme contenente unicamente lo stato x_e). Ovviamente gli insiemi son disgiunti, e dalla seconda condizione del teorema si ricava che la loro unione è tutto l'insieme Δ_{2^n}. Per ogni $x = \delta_{2^n}^h \in \mathcal{S}_{t+1}$, esiste qualche $u = \delta_{2^m}^j$ tale che $L \ltimes \delta_{2^m}^j \ltimes \delta_{2^n}^h \in \mathcal{S}_t$; a questo punto basta porre $\mathrm{Col}_h(K) = \delta_{2^m}^j$, e ripetere questa operazione per ogni $x \in \Delta_{2^n}$, $x \neq x_e$. In questo modo si ottiene la matrice di retroazione voluta K. ♢

Definizione 4.1: L'insieme $\mathcal{S}_t$, $t \in \mathbb{Z}_+$ viene detto t-esima *corona* del sistema, relativa allo stato x_e.

Osservazione 4.1: Come si può dedurre dalla dimostrazione, non è detto che la matrice di retroazione K sia unica: se infatti da uno stesso stato appartenente all'insieme $\mathcal{S}_{t+1}$ si può passare a 2 o più stati distinti di $\mathcal{S}_t$, la colonna di K relativa a quello stato potrà assumere più di un valore. □

Nel caso in cui venga fornita una matrice di retroazione K e si voglia controllare che questa effettivamente sia una matrice stabilizzante per la BCN, è possibile sfruttare la Proposizione 2.5. Questa proposizione si applica alle BN, e in effetti, con la retroazione dallo stato si passa da una BCN ad una BN. Se infatti l'ingresso applicato alla BCN è $u(t) = Kx(t)$, l'evoluzione della rete è descritta da

$$x(t+1) = L \ltimes u(t) \ltimes x(t) = L \ltimes K \ltimes x(t) \ltimes x(t) \, ;$$

analizzando $x(t) \ltimes x(t)$ si può verificare che se $x(t) = \delta_{2^n}^i$, il prodotto risulta un vettore canonico di dimensione 2^{2n} con l'uno posizionato all'indice $(i-1)2^n+i$. Si ha quindi $x(t) \ltimes x(t) = \Phi_{2^n} x(t)$, con $\Phi_{2^n} \in \mathcal{L}^{2^{2n} \times 2^n}$ che presenta la seguente forma:

$$\Phi_{2^n} = \left[\begin{array}{cccc} \delta_{2^{2n}}^1 & \delta_{2^{2n}}^{2^n+2} & \cdots & \delta_{2^{2n}}^{2^{2n}} \end{array} \right] .$$

Il sistema di partenza (3.3) può quindi, quando è retroazionato dallo stato, essere riscritto nel seguente modo:

$$x(t+1) = L \ltimes K \ltimes \Phi_{2^n} \ltimes x(t) .$$

La matrice $\tilde{L} = L \ltimes K \ltimes \Phi_{2^n}$ risulta una matrice logica di dimensioni $2^n \times 2^n$, e quindi il sistema retroazionato evolve secondo

$$x(t+1) = \tilde{L} x(t) ,$$

per cui il tutto è stato ricondotto ad una BN con matrice di evoluzione dello stato $\tilde{L}$.
Sfruttando la Proposizione già citata risulta immediato verificare il seguente

Corollario 4.1 *La BCN (3.3) può essere stabilizzata allo stato $x_e = \delta_{2^n}^i$ grazie alla retroazione dallo stato effettuata con la matrice K se e solo se la matrice*

$$\tilde{L}^{2^n} = (L \ltimes K \ltimes \Phi_{2^n})^{2^n}$$

ha tutte le colonne pari a $\delta_{2^n}^i$.

Esempio 4.1: Si consideri nuovamente la BCN riportata nell'Esempio 3.7. Nella successiva Figura 4.1 sono state evidenziate le diverse corone del sistema rispetto all'unico stato a cui la rete è stabilizzabile, cioè $x_e = \delta_8^2$ (vedere Esempio 3.8). Gli archi effettuati con tratto continuo sono quelli che permettono di passare da una corona alla precedente, mentre quelli a trattini sono i restanti archi del sistema; inoltre ogni arco è stato etichettato con l'ingresso che lo genera. Risulta quindi immediato verificare che la matrice

$$K = \left[\begin{array}{cccccccc} 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{array} \right]$$

permette di stabilizzare il sistema nello stato $x_e = \delta_8^2$, e in realtà la terza, quarta e settima colonna possono presentare anche il vettore δ_2^2 e

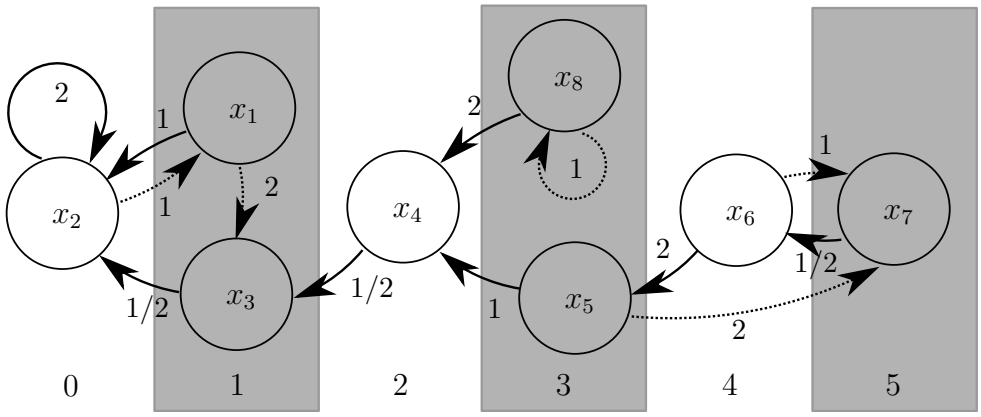


Figura 4.1: Grafo di stato con suddivisione degli stati in corone per la rete dell'Esempio 3.7.

la matrice resta comunque stabilizzante. Nel seguito si calcola la matrice $\tilde{L}$:

$$\tilde{L} = L \ltimes K \ltimes \Phi_8 = \delta_8 \left[\begin{array}{cccccccc} 2 & 2 & 2 & 3 & 4 & 5 & 6 & 4 \end{array} \right] .$$

Effettivamente elevando all'ottava potenza $\tilde{L}$ si ottiene una matrice che presenta tutte le colonne uguali e pari a δ_8^2.
Se invece per esempio si considera una matrice $\tilde{K}$ uguale a K in tutte le colonne tranne che per la quinta che viene posta uguale a δ_2^2, si ottiene la seguente matrice

$$\bar{L} = L \ltimes K \ltimes \Phi_8 = \delta_8 \left[\begin{array}{cccccccc} 2 & 2 & 2 & 3 & 7 & 5 & 6 & 4 \end{array} \right] ,$$

che elevata all'ottava potenza dà

$$\bar{L}^8 = \delta_8 \left[\begin{array}{cccccccc} 2 & 2 & 2 & 2 & 5 & 6 & 7 & 2 \end{array} \right] .$$

Per il Corollario 4.1, $\tilde{K}$ non è una matrice di retroazione stabilizzante, come si può effettivamente notare dal grafo di stato della rete; scegliendo infatti come ingresso δ_2^2 quando lo stato è δ_8^5, si va ad innestare il ciclo composto dai nodi $5, 7, 6$. Questo comporta che se lo stato iniziale è uno di questi tre, con la matrice di retroazione $\tilde{K}$ lo stato è destinato a rimanere confinato in questo sottoinsieme di Δ_8.

♣

Osservazione 4.2: Nell'esempio appena riportato il grafo di stato è risultato lo strumento più immediato per la ricostruzione della matrice di retroazione, ma questo grazie alla dimensione ridotta del numero di

stati della BCN considerata; all'aumentare del numero di stati l'analisi del grafo non risulta più essere un metodo ottimale per la costruzione della matrice K; si deve a questo punto seguire l'idea presentata nella dimostrazione della Proposizione 4.1. □

Osservazione 4.3: Il metodo di calcolo della matrice K qui esposto in realtà non fornisce tutte le possibili matrici di retroazione che stabilizzano il sistema ad un determinato stato x_e, ma fornisce tutte le matrici di retroazione che permettono di stabilizzare il sistema con il minimo numero di passi a partire da ogni stato. A titolo esemplificativo, se nell'ultimo Esempio si pone la prima colonna di K pari a δ_2^2, la matrice K è comunque stabilizzante; la sostanziale differenza è che usando questa matrice di retroazione, se lo stato iniziale è δ_8^1, lo stato di equilibrio si raggiunge in 2 passi, mentre con le matrici trovate nell'Esempio, si raggiunge l'equilibrio in 1 passo a partire da δ_8^1. Visto che il raggiungimento dello stato di equilibrio nel minimo tempo possibile può essere di norma ritenuto un aspetto positivo, è stato trattato solo questo modo di calcolare la matrice di retroazione K. □

Per sfruttare la retroazione dallo stato, è ovviamente necessario essere a conoscenza dello stato che la BCN presenta in un determinato istante. Spesso non è possibile essere a conoscenza di questa informazione, dal momento che il più delle volte si ha a disposizione un'uscita che può dare solo parziale informazione circa lo stato presente assunto dalla rete. Scopo del prossimo paragrafo è quindi lo studio della retroazione dall'uscita.

4.2 Retroazione dall'uscita

Nel caso di controllo di una BCN attraverso la retroazione dall'uscita, l'obiettivo che che ci si pone è quello di stabilizzare ad un determinato stato la rete, conoscendo però unicamente l'uscita del sistema stesso. La differenza rispetto al caso precedente risulta evidente, dal momento che più stati possono indurre la stessa uscita, quindi non è possibile generare un valore di ingresso diverso per ogni stato, ma soltanto un valore di ingresso diverso per ciascun insieme di stati che forniscono lo stesso valore di uscita. La richiesta risulta quindi più impegnativa rispetto al caso di retroazione dallo stato. Ovviamente condizione necessaria affinchè sia possibile la retroazione è che lo stato a cui si vuole stabilizzare il sistema sia raggiungibile da tutti gli altri stati, e inoltre lo stato deve essere un punto fisso per un qualche ingresso della BCN.

In letteratura esiste uno studio recentemente pubblicato, [6], mentre non sono presenti altri lavori a riguardo dell'argomento. Nella prima parte di questo paragrafo si riprende questo studio, che risolve il problema della retroazione dall'uscita ma solo in un caso particolare, per cui il problema si può considerare ancora aperto.
Successivamente si propongono alcune strategie per la soluzione attraverso una retroazione statica e tempo invariante dall'uscita: due di queste strategie a fronte di numerosi calcoli sono in grado di determinare per una data BCN l'esistenza della retroazione dall'uscita voluta e anche la matrice di retroazione, mentre altri due metodi, concettualmente molto simili, cercano la matrice di retroazione che risponda al problema posto procedendo per tentivi, e solo quando tutti i possibili tentativi sono falliti, possono stabilire che non esiste la retroazione dall'uscita voluta. Tutti questi metodi risultano comunque molto dispendiosi da un punto di vista numerico.
Viene poi proposta una retroazione dall'uscita di tipo tempo variante, che, a fronte di alcune condizioni preliminari, fornisce una soluzione al problema proposto e risulta numericamente più efficiente.
Per i metodi esposti sono state create delle funzioni Matlab, che verranno riportate in Appendice A, per ottenere gli ingressi da dare in corrispondenza alle diverse uscite per stabilizzare il sistema allo stato di equilibrio.
Infine si accenna ad una ulteriore soluzione al problema, grazie alla quale la retroazione dall'uscita viene ricondotta alla retroazione dallo stato attraverso la ricostruzione dello stato stesso grazie all'uscita; ovviamente questo metodo è utilizzabile nel caso la rete risulti ricostruibile.

4.2.1 Retroazione tempo invariante minima

Il metodo suggerito in [6] si basa su una retroazione tempo invariante, per cui ad ogni valore dell'uscita corrisponde un valore dell'ingresso che resta sempre lo stesso per tutta la durata del controllo; per questo si può applicare un ragionamento simile a quello usato nella retroazione dallo stato per dimostrare che la BCN retroazionata si riduce ad essere una BN. La retroazione dall'uscita implica infatti che l'ingresso venga calcolato nel seguente modo:

$$u(t) = Ky(t) \,,$$

con $K \in \mathcal{L}^{2^m \times 2^p}$, ma dal momento che $y(t) = Hx(t)$, si ha anche che

$$u(t) = Ky(t) = KHx(t) \,.$$

Il sistema retroazionato è quindi descritto dalla seguente forma algebrica:

$$x(t+1) = L \ltimes K \ltimes H \ltimes x(t) \ltimes x(t) = L \ltimes K \ltimes H \ltimes \Phi_{2^n} \ltimes x(t) \,;$$

per cui la BCN (3.3) diventa una BN con matrice $\tilde{L} = L \ltimes K \ltimes H \ltimes \Phi_{2^n}$ (che è effettivamente una matrice logica di dimensione $2^n \times 2^n$). Sfruttando nuovamente la Proposizione 2.5 si può enunciare il seguente

Teorema 4.1 *La BCN (3.3) può essere stabilizzata allo stato $x_e = \delta_{2^n}^i$ grazie alla retroazione dall'uscita se e solo se esiste una matrice $K \in \mathcal{L}^{2^m \times 2^p}$ tale che*

$$\tilde{L}^{2^n} = (L \ltimes K \ltimes H \ltimes \Phi_{2^n})^{2^n}$$

ha tutte le colonne pari a $\delta_{2^n}^i$.

Questo teorema dà una condizione necessaria e sufficiente per capire se K è una matrice con la quale si può effettuare retroazione dall'uscita, ma non fornisce alcun metodo per ottenerla. Ciononostante, è ovvio che se $u(t) = Ky(t)$ è un ingresso che stabilizza il sistema con retroazione dall'uscita, l'ingresso $u(t) = KHx(t)$ stabilizza il sistema con retroazione dallo stato.
Il contributo dato dal lavoro [6] è quello di determinare una matrice K, se esiste, tale che KH sia una matrice di retroazione minima dallo stato che stabilizza il sistema allo stato $x_e = \delta_{2^n}^i$; il limite del lavoro però risiede nel fatto che limitando la ricerca a una matrice KH di retroazione minima (nel senso spiegato nell'Osservazione 4.3), possono esistere dei casi in cui non è possibile trovare una K per cui questo succeda, pur esistendo una retroazione dall'uscita K stabilizzante, per la quale non viene rispettato il vincolo che KH sia una matrice di retroazione stabilizzante minima dallo stato. Perciò questo tipo di retroazione dall'uscita tempo invariante è stata definita a sua volta minima.
Nel seguito sono riportati i diversi passaggi per la costruzione della matrice K, e con un controesempio si evidenzia il limite del lavoro, che comunque risulta efficace nelle reti Booleane di controllo alle quali è applicabile.
Riprendendo la dimostrazione della Proposizione 4.1, per ogni stato $x = \delta_{2^n}^j \in \Delta_{2^n}$, che appartiene ad una determinata corona $\mathcal{S}_k$, esiste un insieme P_j formato dai possibili $u \in \Delta_{2^m}$ grazie ai quali è possibile passare dallo stato $\delta_{2^n}^j$, ad uno stato della corona $\mathcal{S}_{k-1}$ (nel caso sia in esame lo stato $x_e = \delta_{2^n}^i$ a cui si vuole stabilizzare il sistema, P_i è composto da tutti gli ingressi per i quali x_e è punto fisso).
Ovviamente l'insieme

$$\Lambda = \{G = \delta_{2^m} \left[\begin{array}{ccc} p_1 & \cdots & p_{2^n} \end{array} \right] \mid p_i \in P_i, i = 1, \ldots, 2^n\}$$

contiene tutte le matrici con cui è possibile fare retroazione (minima) dallo stato.

Definendo l'insieme

$$\Theta = \{K = \delta_{2^m}\left[\begin{array}{ccc} v_1 & \cdots & v_{2^p}\end{array}\right] \mid KH \in \Lambda\},$$

si ha ovviamente che K è una matrice grazie alla quale è possibile fare retroazione dall'uscita. Per stabilire l'esistenza di K ed eventualmente calcolarla, si definiscono inizialmente gli insiemi $\mathcal{T}_k$, $k = 1, \ldots, 2^p$, ciascuno dei quali contiene tutti gli stati $\delta_{2^n}^j$ che danno per uscita $\delta_{2^p}^k$, cioè

$$\mathcal{T}_k = \{\delta_{2^n}^j \mid \mathrm{Col}_j(H) = \delta_{2^p}^k\}.$$

Ovviamente i diversi $\mathcal{T}_k$ formano una partizione dell'insieme degli stati Δ_{2^n}. Per ogni $\mathcal{T}_k$, si può costruire un ulteriore insieme $I(k)$, così definito:

$$I(k) = \begin{cases} \bigcap_{j \mid \delta_{2^n}^j \in \mathcal{T}_k} P_j, & \mathcal{T}_k \neq \varnothing, \\ \{1, 2, \ldots, 2^m\}, & \mathcal{T}_k = \varnothing. \end{cases}$$

Sfruttando questi ultimi insiemi introdotti è possibile enunciare il seguente

Teorema 4.2 *Il sistema è globalmente stabilizzabile a $x_e = \delta_{2^n}^i$ grazie ad un controllo in retroazione dall'uscita $u(t) = Ky(t)$, $K \in \Theta$, se*

$$I(k) \neq \varnothing,\ 1 \leq k \leq 2^p. \tag{4.1}$$

Dimostrazione: Se (4.1) vale, si può costruire la matrice

$$K = \delta_{2^m}\left[\begin{array}{ccc} v_1 & \ldots & v_{2^p}\end{array}\right],\ v_k \in I(k).$$

Se $H = \delta_{2^p}\left[\begin{array}{ccc} w_1 & \ldots & w_{2^n}\end{array}\right]$, si ottiene

$$KH = \delta_{2^m}\left[\begin{array}{ccc} v_{w_1} & \ldots & v_{w_{2^n}}\end{array}\right].$$

Dal momento che $v_{w_i} \in I(w_i) \subseteq P_i$, $\forall\, i = 1, \ldots, 2^n$, allora $K \in \Theta$, per cui KH è una matrice di retroazione dallo stato, e K è matrice di retroazione dall'uscita. $\diamondsuit$

Osservazione 4.4: In [6] la condizione del Teorema precedente non è riferita solo come sufficiente, ma anche come necessaria; ciò comporterebbe che, se una BCN ha $I(k) = \varnothing$ per qualche $1 \leq k \leq 2^p$, allora non si può trovare una retroazione dall'uscita che stabilizzi la BCN a x_e. Questa conclusione in generale non è vera, dal momento che si possono trovare dei controesempi (vedere Esempio 4.3) e nel lavoro citato non sono evidenziate in maniera chiara le condizioni di minimalità di convergenza ad

x_e che soggiaciono alla necessità (anche se nelle conclusioni si può forse intendere qualcosa di simile, laddove si segnala l'opportunità di un ulteriore lavoro dedicato alla ricerca delle matrici di retroazione dallo stato stabilizzanti). □

Esempio 4.2: Data la BCN determinata dalle seguenti matrici (L viene assegnata attraverso le sottomatrici che la compongono)

$$L_1 = \delta_8 \left[\begin{array}{cccccccc} 1 & 7 & 6 & 1 & 7 & 8 & 7 & 2 \end{array} \right] \qquad L_2 = \delta_8 \left[\begin{array}{cccccccc} 8 & 3 & 3 & 1 & 4 & 6 & 1 & 8 \end{array} \right]$$

$$L_3 = \delta_8 \left[\begin{array}{cccccccc} 1 & 8 & 3 & 4 & 4 & 6 & 1 & 5 \end{array} \right] \qquad L_4 = \delta_8 \left[\begin{array}{cccccccc} 2 & 4 & 6 & 3 & 7 & 2 & 1 & 8 \end{array} \right]$$

$$H = \delta_4 \left[\begin{array}{cccccccc} 1 & 1 & 2 & 2 & 3 & 3 & 4 & 4 \end{array} \right]$$

si cercano le possibili matrici di retroazione dall'uscita del sistema tali che lo stato a cui viene stabilizzato il sistema sia $x_e = \delta_{2^n}^1$, a cui la rete è controllabile e che risulta anche punto fisso.
Analizzando allora L_{tot} e le sue potenze si può studiare la divisione degli stati in corone, che è la seguente[2]:

$$\mathcal{S}_0 = \{1\};\ \mathcal{S}_1 = \{7, 4\};\ \mathcal{S}_2 = \{2, 5\};\ \mathcal{S}_3 = \{6, 8\};\ \mathcal{S}_4 = \{3\}\ .$$

A questo punto studiando L si possono trovare i diversi P_i che risultano

$$P_1 = \{1, 3\};\ P_2 = \{1, 4\};\ P_3 = \{1, 4\};\ P_4 = \{1, 2\};\ P_5 = \Delta_4;$$

$$P_6 = \{2, 4\};\ P_7 = \{2, 3, 4\};\ P_8 = \{1, 3\}\ .$$

I diversi insiemi $I(k)$ risultano quindi:

$$I(1) = \{1\};\ I(2) = \{1\};\ I(3) = \{2, 4\};\ I(4) = \{3\}\ .$$

Individuati questi insiemi, grazie al Teorema 4.2 e alla sua dimostrazione, si può concludere che

$$K_1 = \delta_4 \left[\begin{array}{cccc} 1 & 1 & 2 & 3 \end{array} \right] \quad \text{e} \quad K_2 = \delta_4 \left[\begin{array}{cccc} 1 & 1 & 4 & 3 \end{array} \right]\ ,$$

sono due matrici di retroazione dall'uscita stabilizzanti. ♣

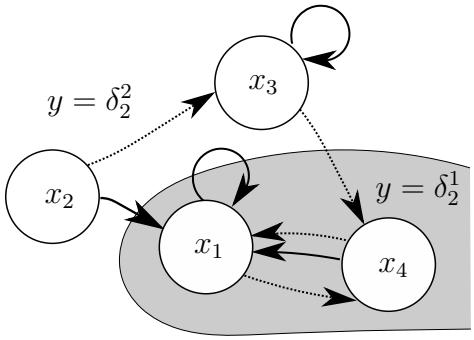
Esempio 4.3: La semplice BCN qui riportata è un controesempio per dimostrare che il Teorema precedente non è in grado di individuare tutte

[2]per semplicità notazionale, in questi insiemi si indica semplicemente l'indice del relativo vettore canonico. Per esempio il valore 1 che contenuto nell'insieme $\mathcal{S}_0$ è un modo abbreviato per indicare che il vettore δ_8^1 è contenuto in $\mathcal{S}_0$. Questa notazione semplificata verrà anche usata in altri Esempi, per rendere più scorrevole la lettura.

le retroazioni dall'uscita tempo invarianti che permettono la stabilizzazione del sistema. La BCN presenta le seguenti matrici L e H, e la sua rappresentazione in grafo di stato è riportata a lato.

$$L = \delta_4 \left[\begin{array}{cccc|cccc} 1 & 1 & 3 & 1 & 4 & 3 & 4 & 1 \end{array} \right]$$

$$H = \delta_2 \left[\begin{array}{cccc} 1 & 2 & 2 & 1 \end{array} \right]$$



Sono rappresentati a tratto continuo rappresentano gli archi relativi all'ingresso δ_2^1, mentre sono rappresentati a tratteggio gli archi relativi a δ_2^2. Lo stato rispetto al quale si vuole stabilizzare il sistema è $x_e = \delta_4^1$, che è raggiungibile da tutti gli stati ed è anche punto fisso relativamente all'ingresso δ_2^1. La suddivisione in corone degli stati è la seguente:

$$\mathcal{S}_0 = \{1\}; \quad \mathcal{S}_1 = \{2, 4\}; \quad \mathcal{S}_3 = \{3\} \, ,$$

mentre i diversi P_i sono:

$$P_1 = \{1\}; \quad P_2 = \{1\}; \quad P_3 = \{2\}; \quad P_4 = \{1, 2\}.$$

I due insiemi $I(k)$ risultano quindi

$$I(1) = \{1\}; \quad I(2) = \varnothing \, .$$

Se il Teorema 4.2 fornisse anche una condizione necessaria per la stabilizzazione, non esisterebbe una retroazione dall'uscita stabilizzante allo stato x_e. Tuttavia se si prende

$$K = \delta_2 \left[\begin{array}{cc} 1 & 2 \end{array} \right] \, ,$$

dal grafo di stato risulta evidente che con questa retroazione dall'uscita il sistema si porta sullo stato di equilibrio desiderato. Con questa retroazione dall'uscita, il tempo richiesto per raggiungere lo stato di equilibrio a partire dallo stato δ_4^2 è pari a 3 passi, mentre nel caso della retroazione dallo stato, il tempo è pari a 1 passo, e quindi il tempo è maggiore nella retroazione dall'uscita rispetto alla retroazione dallo stato.
È stato quindi trovato un controesempio che dimostra che la condizione del Teorema 4.2 è sufficiente ma non necessaria affinchè esista una retroazione statica dall'uscita. ♣

Osservazione 4.5: Utilizzando lo stesso metodo appena esposto, è anche possibile individuare tutte le retroazioni minime dallo stato di una data BCN, che stabilizzano ad un determinato stato di equilibrio; con lo stesso procedimento infatti, se si prende come matrice stato-uscita

$$H = \delta_{2^n} \left[\begin{array}{cccc} 1 & 2 & \cdots & 2^n \end{array} \right] ,$$

è facile convincersi che le matrici di retroazione che si ottengono sono quelle che permettono la retroazione minima dallo stato. □

4.2.2 Retroazione tempo invariante

Per descrivere i metodi che si introducono in questo contesto, conviene introdurre le seguente ipotesi:

- lo stato a cui si vuole stabilizzare il sistema è lo stato $\delta_{2^n}^1$;
- l'uscita relativa allo stato di equilibrio è $\delta_{2^p}^1$;
- gli stati sono numerati in modo tale che gli stati che danno come uscita $\delta_{2^p}^i,\ i = 1, \ldots, 2^p - 1$, precedono tutti gli stati che danno come uscita $\delta_{2^p}^j,\ j = i+1, \ldots, 2^p$.

Ovviamente queste ipotesi non sono restrittive, perchè è sufficiente un riordino degli stati e delle uscite per soddisfarle.
Si introduce quindi la seguente partizione degli stati, determinata dagli insiemi $\mathcal{T}_i,\ i = 1, \ldots, 2^p$:

$$\mathcal{T}_i = \{\delta_{2^n}^j \ \in \ \Delta_{2^n} \mid H\delta_{2^n}^j = \delta_{2^p}^i\} ,$$

che coincide alla partizione introdotta nel paragrafo precedente. Se la BCN è formata dai sottosistemi $L_j,\ j = 1, \ldots, 2^m$, allora, grazie alla numerazione degli stati adottata, si ha che ogni matrice L_j risulta a sua volta partizionata in sottoblocchi colonna nella seguente maniera:

$$L_j = \left[\begin{array}{c|c|c|c} L_{j,1} & L_{j,2} & \cdots & L_{j,2^p} \end{array} \right] \ j = 1, \ldots, 2^m .$$

Il primo blocco è relativo agli stati che danno come uscita $\delta_{2^n}^1$ e così via; inoltre i sottoblocchi colonna $L_{j,k}$ sono vuoti per $j = 1, 2, \ldots, 2^m$ se nessuno stato fornisce come uscita $\delta_{2^p}^k$. Con la retroazione dall'uscita, come già evidenziato in precedenza, ogni qualvolta il sistema si troverà in uno qualsiasi degli stati appartenenti all'insieme $\mathcal{T}_i$, l'ingresso risulterà

assumere sempre lo stesso valore. Si ha quindi che, scelta la retroazione dall'uscita, la BCN si trasforma in una BN con matrice di transizione

$$\tilde{L} = \left[\; L_{j_1,1} \; \middle| \; L_{j_2,2} \; \middle| \; \cdots \; \middle| \; L_{j_{2^p},2^p} \; \right]$$

con $j_i \in \{1, \ldots, 2^m\}$.
Osservazione 4.6: La matrice $\tilde{L}$ qui introdotta è la stessa matrice che si ottiene con lo svolgimento dei calcoli $\tilde{L} = L \ltimes K \ltimes H \ltimes \Phi_{2^n}$, quando la numerazione degli stati è quella presa in considerazione. □
Ovviamente la retroazione stabilizza il sistema a x_e se e solo se $\tilde{L}$ elevato alla potenza 2^n presenta tutte colonne pari a $\delta^1_{2^n}$.

Osservazione 4.7: Per semplicità di notazione e per una maggiore scorrevolezza nella lettura, nel seguito la dimensione degli stati, 2^n, verrà denotata con N, la dimensione degli ingressi, 2^m, verrà denotata con M, e quella delle uscite, 2^p, con P, soprattutto nel caso siano usate come pedici o apici nelle formule. □

Dopo questa introduzione generale si riporta un primo metodo per la risoluzione del problema, che si basa sull'uso di polinomi in $2^p 2^m$ indeterminate.
Considerata la matrice polinomiale

$$\mathscr{L} = \left[\; \sum_{j_1=1}^{M} L_{j_1,1} z_{j_1,1} \; \middle| \; \sum_{j_2=1}^{M} L_{j_2,2} z_{j_2,2} \; \middle| \; \cdots \; \middle| \; \sum_{j_P=1}^{M} L_{j_P,P} z_{j_P,P} \; \right] ,$$

questa risulta essere composta da polinomi lineari omogenei nelle $2^p 2^m$ indeterminate $z_{j_\nu,\nu}$, $j_\nu = 1, \ldots, 2^m$, $\nu = 1, \ldots, 2^p$.
Risulta abbastanza evidente la somiglianza di $\mathscr{L}$ con la matrice L_{tot} introdotta nel capitolo 3; in effetti, preso in considerazione lo stato $\delta^j_{2^n} \in \mathcal{T}_\nu$, se $[\mathscr{L}]_{ij} = 0$, allora non esiste nessun ingresso che permetta di passare in un passo dallo stato $\delta^j_{2^n}$ a $\delta^i_{2^n}$, mentre se $[\mathscr{L}]_{ij} = z_{j_{\nu_1},\ell} + \cdots + z_{j_{\nu_k},\ell}$, $1 \leq j_{\nu_1} < \cdots < j_{\nu_k} \leq 2^m$, $1 \leq k \leq 2^m$, allora con gli ingressi $\delta^{j_{\nu_1}}_{2^m}, \ldots, \delta^{j_{\nu_k}}_{2^m}$ è possibile il passaggio tra $\delta^j_{2^n}$ e $\delta^i_{2^n}$ (e il valore dell'uscita determinato dallo stato $\delta^j_{2^n}$ è $\delta^\ell_{2^p}$). Questa matrice fornisce quindi una informazione in più rispetto a quella data da L_{tot}, cioè quale sia l'ingresso che permette il passaggio.
Considerando allora la potenza k-esima di $\mathscr{L}$, gli elementi nulli indicano che non esiste una sequenza di ingressi che permetta di passare da uno stato all'altro in k passi, mentre se l'elemento è diverso da 0, allora esiste una sequenza di ingressi di lunghezza k che permette il passaggio e automaticamente si conoscono anche gli ingressi necessari per il passaggio (ma non il loro ordine, a meno di non compiere la moltiplicazione con un

metodo appropriato).
Osservazione 4.8: Dal momento che in ogni insieme $\mathcal{T}_i$, $i = 1, \ldots, 2^p$ l'ingresso indotto dalla retroazione è sempre uguale nel tempo per ogni stato che appartiene a questo insieme, si può sfruttare la seguente considerazione: se un elemento della matrice $\mathscr{L}^k$ contiene un monomio che presenta il prodotto tra due indeterminate $z_{j_{i_1},i}$ e $z_{j_{i_2},i}$, con $j_{i_1} \neq j_{i_2}$, si può ignorare tranquillamente quest'ultimo monomio, poiché attraverso la retroazione in considerazione non si può cambiare l'ingresso scelto per l'insieme $\mathcal{T}_i$. Questa considerazione permette di effettuare meno calcoli. □
Calcolata la potenza k-esima di $\mathscr{L}$, se si considera solo un insieme di 2^p indeterminate $z_{j_1,1}, \ldots, z_{j_P,P}$, $1 \leq j_i \leq 2^m$, e si pongono tutte le altre indeterminate a 0, ogni colonna risulta essere monomia di grado k. Questo risulta evidente se si pensa di operare questa sostituzione direttamente in $\mathscr{L}$ e poi di effettuare l'elevamento a potenza (il risultato finale è ovviamente coincidente); la sostituzione effettuata coincide con il prendere la matrice polinomiale corrispondente alla matrice $\tilde{L}$ ottenuta considerando come retroazione per gli stati appartenenti a $\mathcal{T}_i$ l'ingresso $\delta_{2^m}^{j_i}$, cioè coincide con il prendere la matrice

$$\left[\; L_{j_1,1} z_{j_1,1} \;\middle|\; \cdots \;\middle|\; L_{j_P,P} z_{j_P,P} \; \right] .$$

Quest'ultima è una matrice con colonne monomie di grado 1, e la sua potenza k-esima presenta colonne monomie di grado k nelle 2^p indeterminate $z_{j_1,1}, \ldots, z_{j_P,P}$.
A questo punto è possibile enunciare il seguente

Teorema 4.3 *La BCN (3.3) può essere stabilizzata allo stato x_e attraverso una retroazione statica dall'uscita se e solo se esiste un insieme di 2^p indeterminate $z_{j_1,1}, \ldots, z_{j_P,P}$, $1 \leq j_i \leq 2^m$ tale che $[\mathscr{L}^{2^n}]_{1,1}$ presenta come monomio $z_{j_1,1}^{2^n}$, e tutti gli altri elementi della prima riga di $\mathscr{L}^{2^n}$ presentano in queste indeterminate monomi di grado 2^n.*
Per ogni insieme di inderminate per cui il teorema è verificato, la matrice

$$K = \left[\; \delta_{2^m}^{j_1} \quad \delta_{2^m}^{j_2} \quad \cdots \quad \delta_{2^m}^{j_P} \; \right]$$

è una matrice di retroazione dall'uscita stabilizzante allo stato x_e.

Dimostrazione: Se la BCN è stabilizzabile con una retroazione dall'uscita allo stato $x_e = \delta_{2^n}^1$, per prima cosa deve esistere un ingresso $\delta_{2^m}^{j_1}$ per il quale lo stato x_e è punto fisso, quindi $[\mathscr{L}^{2^n}]_{1,1}$ deve presentare come monomio $z_{j_1,1}^{2^n}$ perchè x_e è raggiungibile da se stesso con l'ingresso $\delta_{2^m}^{j_1}$;

tutti gli altri stati poi devono poter raggiungere lo stato x_e in al massimo 2^n passi, rispettando però il vincolo per il quale tutti gli stati che appartengono a uno stesso $\mathcal{T}_i$ "subiscono" lo stesso ingresso $\delta_{2^m}^{j_i}$. In $\mathscr{L}$ questo si riflette sull'esistenza di almeno un insieme di 2^p indeterminate $z_{j_1,1},\ldots,z_{j_P,P}$, $1 \leq j_i \leq 2^m$, grazie alle quali è possibile trovare in ogni elemento di $[\mathscr{L}^{2^n}]_{1,k}$, $k = 1,\ldots,2^n$ un monomio in queste indeterminate di grado 2^n.
Per l'implicazione inversa, se si considera un insieme delle indeterminate $z_{j_1,1},\ldots,z_{j_P,P}$, $1 \leq j_i \leq 2^m$ che verifica la condizione del teorema, e si pongono a zero tutte le altre in $\mathscr{L}^{2^n}$, si ottiene una matrice con colonne monomie di grado 2^n con l'elemento diverso da 0 in prima posizione per ogni colonna. Come è già stato sottolineato, lo stesso risultato si ottiene se si considera la potenza 2^n-esima di

$$\left[\ L_{j_1,1}z_{j_1,1} \,\middle|\, \cdots \,\middle|\, L_{j_P,P}z_{j_P,P}\ \right] .$$

Risulta quindi ovvio che la matrice

$$\tilde{L} = \left[\ L_{j_1,1} \,\middle|\, \cdots \,\middle|\, L_{j_P,P}\ \right]$$

elevata alla potenza 2^n-esima presenta tutte le colonne pari a $\delta_{2^n}^1$, e quindi la BCN è stabilizzabile allo stato x_e tramite retroazione dall'uscita.
Il fatto che la matrice K sia la matrice di retroazione dall'uscita cercata è una conseguenza immediata della dimostrazione. $\diamondsuit$

Il teorema appena riportato fornisce quindi una condizione necessaria e sufficiente affinchè sia possibile trovare una retroazione statica dall'uscita, ed ha quindi validità più generale del Teorema 4.2, fatto che si può subito vedere con i successivi esempi.

Esempio 4.4: Si riprende inizialmente l'Esempio 4.3, per dimostrare che con il teorema introdotto in questa sezione è possibile trovare una retroazione dall'uscita che prima non era individuabile. Per sfruttare il teorema è necessario un riordino degli stati per ottenere la numerazione riportata all'inizio della sezione; una permutazione che permette di ottenere la numerazione voluta è

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 2 & 3 \end{pmatrix} .$$

Con questa permutazione, la matrice di transizione L e la matrice stato-uscita diventano le seguenti:

$$L = \delta_4 \left[\ 1 \quad 1 \,\middle|\, 1 \quad 4 \,\middle\|\, 2 \quad 1 \,\middle|\, 4 \quad 2\ \right]$$

$$H = \delta_2 \left[\begin{array}{cccc} 1 & 1 & 2 & 2 \end{array} \right] .$$

Il grafo di stato della BCN con la nuova numerazione è riportato in Figura 4.2. La matrice $\mathscr{L}$ risulta quindi la seguente

$$\mathscr{L} = \left[\begin{array}{cccc} z_{1,1} & z_{1,1} + z_{2,1} & z_{1,2} & 0 \\ z_{2,1} & 0 & 0 & z_{2,2} \\ 0 & 0 & 0 & 0 \\ & 0 & z_{2,2} & z_{1,2} \end{array} \right] ,$$

la cui potenza quarta potenza presenta la seguente prima riga:

$$\left[\begin{array}{cccc} z_{1,1}^4 + z_{2,1}^4 & z_{1,1}^4 & z_{1,1}^3 z_{1,2} + z_{1,1}^2 z_{2,2}^2 & z_{1,1}^3 z_{2,2} + z_{2,1}^3 z_{2,2} \end{array} \right] .$$

Risulta evidente che l'unica coppia di indeterminate che soddisfa le condizioni del teorema è $z_{1,1}, z_{2,2}$, per cui con la matrice di retroazione dall'uscita

$$K = \delta_2 \left[\begin{array}{cc} 1 & 2 \end{array} \right]$$

è possibile stabilizzare il sistema allo stato $x_e = \delta_4^1$. A differenza del metodo usato precedentemente, con questo si è in grado di individuare la retroazione dall'uscita. ♣

Esempio 4.5: Per studiare l'Esempio 4.2 con questa nuova tecnica, si introduce una notazione più agevole: al posto di indicare l'indeterminata con $z_{j_i,i}$, si usa la notazione (j_i, i) che rende la scrittura e la lettura più veloce. Nel seguito si riportano solo gli insiemi di indeterminate che soddisfano il teorema, e la relativa matrice di retroazione K:

$$\begin{array}{rlll}
\mathbf{1} : \{(1,1),(1,2),(4,3),(3,4)\} & \Rightarrow & K_1 = \delta_4 \left[\begin{array}{cccc} 1 & 1 & 4 & 3 \end{array} \right] ; \\
\mathbf{2} : \{(1,1),(1,2),(2,3),(3,4)\} & \Rightarrow & K_2 = \delta_4 \left[\begin{array}{cccc} 1 & 1 & 2 & 3 \end{array} \right] ; \\
\mathbf{3} : \{(1,1),(1,2),(1,3),(3,4)\} & \Rightarrow & K_3 = \delta_4 \left[\begin{array}{cccc} 1 & 1 & 1 & 3 \end{array} \right] ; \\
\mathbf{4} : \{(1,1),(4,2),(4,3),(3,4)\} & \Rightarrow & K_4 = \delta_4 \left[\begin{array}{cccc} 1 & 4 & 4 & 3 \end{array} \right] ; \\
\mathbf{5} : \{(1,1),(4,2),(1,3),(3,4)\} & \Rightarrow & K_5 = \delta_4 \left[\begin{array}{cccc} 1 & 4 & 1 & 3 \end{array} \right] ; \\
\mathbf{6} : \{(3,1),(1,2),(2,3),(3,4)\} & \Rightarrow & K_6 = \delta_4 \left[\begin{array}{cccc} 3 & 1 & 2 & 3 \end{array} \right] ; \\
\mathbf{7} : \{(3,1),(1,2),(1,3),(3,4)\} & \Rightarrow & K_7 = \delta_4 \left[\begin{array}{cccc} 3 & 1 & 1 & 3 \end{array} \right] ; \\
\mathbf{8} : \{(3,1),(4,2),(1,3),(3,4)\} & \Rightarrow & K_8 = \delta_4 \left[\begin{array}{cccc} 3 & 4 & 1 & 3 \end{array} \right] ; \\
\mathbf{9} : \{(3,1),(1,2),(4,3),(3,4)\} & \Rightarrow & K_9 = \delta_4 \left[\begin{array}{cccc} 3 & 1 & 4 & 3 \end{array} \right] ; \\
\mathbf{10} : \{(3,1),(4,2),(4,3),(3,4)\} & \Rightarrow & K_{10} = \delta_4 \left[\begin{array}{cccc} 3 & 4 & 4 & 3 \end{array} \right] .
\end{array}$$

Le prime 2 matrici di retroazione sono quelle già trovate nell'Esempio 4.2, mentre le successive 8 non erano state individuate; si può facilmente verificare che sono tutte matrici di retroazione e che sono anche le uniche compatibili con una retroazione statica dall'uscita. ♣

Un altro metodo grazie al quale si può stabilire l'esistenza o meno di una retroazione statica dall'uscita per una BCN, si basa sullo studio dei cicli della stessa. Dal momento che attraverso la retroazione statica si passa da una BCN a una BN, si sa a priori che se la BN che si ottiene non presenta unicamente un ciclo di lunghezza 1 in corrispondenza allo stato x_e, allora la retroazione utilizzata non è stabilizzante allo stato desiderato. Se quindi si vuole ottenere una retroazione stabilizzante, bisogna fare in modo che con gli ingressi che si utilizzano non sia mai possibile che si attivi uno dei cicli della BCN.
Considerando un ciclo di periodo $T+1$ di una BCN, questo può essere scritto, come già evidenziato nel Capitolo 3, nel seguente modo:

$$\mathscr{C}_i = (x(t), u(t)), (x(t+1), u(t+1)), \ldots, (x(t+T), u(t+T)) \,,$$

con $L \ltimes u(t+T) \ltimes x(t+T) = x(t+T+1) = x(t)$.
Visto che la retroazione applicata è dall'uscita, è chiaro che di tutti i cicli compatibili con la mappa ingresso-stato della BCN, quelli che possono effettivamente crearsi a seguito della retroazione dall'uscita sono tutti e soli i $\mathscr{C}_i$ per i quali per ogni coppia $x(t+h), x(t+k)$, $h \neq k$, $h, k = 0, \ldots, T$ di stati che appartengono a $\mathscr{C}_i$ e che appartengono anche allo stesso $\mathcal{T}_j$, $1 \leq j \leq 2^p$, si ha $u(t+h) = u(t+k)$.
Se si escludono dai cicli in esame

- tutti quelli di lunghezza 1 che coinvolgono lo stato di equilibrio x_e;
- tutti quelli che per gli stati che appartengono a $\mathcal{T}_1$ necessitano di un ingresso per il quale x_e non è di equilibrio (ovviamente l'ingresso scelto per gli stati di $\mathcal{T}_1$ deve lasciare x_e invariato, altrimenti non si riesce a stabilizzare il sistema)

i cicli che rimangono sono quelli che hanno possibilità di nascere grazie ad una retroazione statica dall'uscita, e la cui eventuale presenza è ovviamente indesiderata. Il loro insieme (cioè quello che contiene tutti i cicli che devono essere evitati) viene denominato $\mathscr{N}$. Per individuare l'eventuale retroazione stabilizzante dall'uscita, che può anche non essere unica, è necessario trovare per ogni $\mathcal{T}_j$ un ingresso u_j in modo da non attivare nessuno dei cicli in esame.

Preso in analisi un ciclo di periodo T, $\mathscr{C}_i \in \mathscr{N}$, questo contiene T stati che appartengono a $\mathcal{T}_{j_1}, \ldots, \mathcal{T}_{j_k}$, $1 \leq k \leq \max\{T, 2^p\}$, ed è generato dagli ingressi $\bar{u}_{j_1}, \ldots, \bar{u}_{j_k}$; per non attivare questo ciclo, detti $u_1, \ldots, u_{2^p}$ gli ingressi scelti per la retroazione rispettivamente per le uscite $\delta^1_{2^p}, \ldots, \delta^{2^p}_{2^p}$, la retroazione non può presentare contemporaneamente $u_{j_1} = \bar{u}_{j_1}, \ldots, u_{j_k} = \bar{u}_{j_k}$, altrimenti il ciclo si attiva.
Si può quindi introdurre la seguente funzione $\wedge_c : \Delta_{2^m} \times \Delta_{2^m} \to \mathcal{B}$ tale che

$$u_i \wedge_c u_j = 1 \Leftrightarrow [u_i]_k = [u_j]_k, \; \forall\, k = 1, \ldots, 2^m \, ;$$

dove con $[u_i]_k$ si intende l'elemento k-esimo del vettore $u_i \in \Delta_{2^m}$. Grazie a questa funzione, la condizione per cui il ciclo $\mathscr{C}_i$ non si attivi può essere riscritta nella seguente maniera:

$$(u_{j_1} \wedge_c \bar{u}_{j_1}) \wedge (u_{j_2} \wedge_c \bar{u}_{j_2}) \wedge \cdots \wedge (u_{j_k} \wedge_c \bar{u}_{j_k}) = 0 \, , \tag{4.2}$$

con $\wedge$ normale operatore logico AND.
A questo punto è possibile enunciare il seguente

Teorema 4.4 *La BCN (3.3) è stabilizzabile attraverso una retroazione statica dall'uscita se e solo se esiste un insieme di 2^p elementi $u_1, \ldots, u_p, u_i \in \Delta_{2^m}$, in grado di soddisfare (4.2) per ogni ciclo $\mathscr{C}_i \in \mathscr{N}$, e con u_1 ingresso di equilibrio per x_e. La matrice*

$$K = \begin{bmatrix} u_1 & u_2 & \cdots & u_{2^p} \end{bmatrix}$$

permette la retroazione dall'uscita cercata.

Dimostrazione: Se la BCN è stabilizzabile attraverso retroazione statica dall'uscita, esiste una scelta degli ingressi $u_1, \ldots, u_{2^p}$ da associare ai rispettivi insiemi $\mathcal{T}_1, \ldots, \mathcal{T}_{2^p}$ in modo che la BN che si trova applicando la retroazione abbia un solo ciclo in x_e. Questo comporta che gli ingressi soddisfano l'equazione (4.2) per ogni ciclo $\mathscr{C}_i \in \mathscr{N}$, altrimenti uno di questi sarebbe attivo e la BN presenterebbe più di un ciclo. Inoltre l'ingresso u_1 deve ovviamente lasciare x_e invariato.
Se viceversa si verifica la condizione sugli ingressi, la BN che si viene a creare con la retroazione dall'uscita effettuta con questi ingressi presenta un solo ciclo di lunghezza 1 in x_e, e quindi la BCN è ovviamente stabilizzabile proprio grazie a questa retroazione. $\diamondsuit$

Nel seguito si ripropongono gli stessi esempi visti in precedenza, applicando però la nuova tecnica dei cicli per ottenere la matrice di retroazione dallo stato.

Esempio 4.6: Nel primo esempio si applica il nuovo procedimento sulla BCN presentata nell'Esempio 4.4, utilizzando la nuova numerazione degli stati lì introdotta. Per una maggiore comprensione si riporta in Figura 4.2 il grafo di stato che si ottiene con la nuova numerazione degli stati.

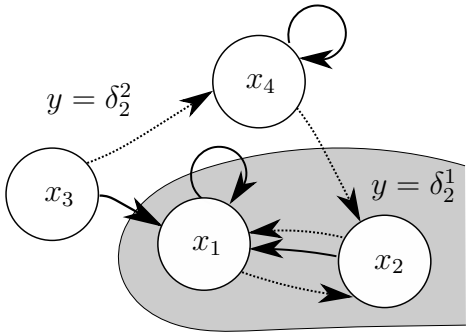


Figura 4.2: Grafo di stato relativo alla BCN dell'esempio 4.6.

La suddivisione degli stati a seconda dell'uscita che producono è ovviamente la seguente:

$$\mathcal{T}_1 = \{\delta_4^1, \delta_4^2\} \qquad \mathcal{T}_2 = \{\delta_4^3, \delta_4^4\} \, .$$

I cicli che compaiono in questa BCN sono i seguenti quattro:

$\mathscr{C}_1 = (\delta_4^1, \delta_2^1)$; $\mathscr{C}_2 = (\delta_4^4, \delta_2^1)$; $\mathscr{C}_3 = ((\delta_4^1, \delta_2^2), (\delta_4^2, \delta_2^1))$; $\mathscr{C}_4 = ((\delta_4^1, \delta_2^2), (\delta_4^2, \delta_2^2))$.

Tra questi, i cicli che appartengono all'insieme $\mathscr{N}$ sono $\mathscr{C}_2$ e $\mathscr{C}_4$, quindi le 2 equazioni che vanno risolte sono:

$$u_2 \wedge_c \delta_2^1 = 0 \, ; \qquad u_1 \wedge_c \delta_2^2 = 0 \, ;$$

da cui si ottiene facilmente che $u_2 = \delta_2^2$, e $u_1 = \delta_2^1$. L'ingresso preso per l'uscita δ_2^1 assicura anche che $x_e = \delta_2^1$ sia punto fisso della BN che si ottiene retroazionando, e quindi le condizioni del Teorema 4.4 sono verificate. La matrice di retroazione stabilizzante allo stato x_e è quindi

$$K = \delta_2 \left[\begin{array}{cc} 1 & 2 \end{array} \right] \, .$$

Il risultato ottenuto per la matrice di retroazione statica stabilizzante dall'uscita è ovviamente identico a quello ottenuto nell'Esempio 4.4. ♣

Esempio 4.7: Si applica qui il metodo dei cicli alla BCN dell'Esempio 4.2. La BCN in esame presenta 168 cicli distinti, che per ragione di

spazio non vengono qui riportati, ma l'analisi dei cicli appartenenti a $\mathscr{N}$ riduce drasticamente il numero dei cicli da controllare, portandolo a 9; nel seguito vengono riportati i cicli appartenenti a questo insieme:

$$\mathscr{C}_1 = (\delta_8^3, \delta_4^2)\,; \quad \mathscr{C}_2 = (\delta_8^3, \delta_4^3)\,; \quad \mathscr{C}_3 = (\delta_8^4, \delta_4^3)\,;$$

$$\mathscr{C}_4 = (\delta_8^6, \delta_4^3)\,; \quad \mathscr{C}_5 = (\delta_8^7, \delta_4^1)\,; \quad \mathscr{C}_6 = (\delta_8^8, \delta_4^2)\,; \quad \mathscr{C}_7 = (\delta_8^8, \delta_4^4)\,;$$

$$\mathscr{C}_8 = \big((\delta_8^2, \delta_4^3), (\delta_8^8, \delta_4^1)\big)\ ; \quad \mathscr{C}_9 = \big((\delta_8^3, \delta_4^4), (\delta_8^6, \delta_4^2), (\delta_8^5, \delta_4^2), (\delta_8^4, \delta_4^4)\big)\ .$$

Gli ingressi per cui $x_e = \delta_8^1$ è punto fisso risultano δ_4^1 e δ_4^3. Le equazioni sugli ingressi che si ottengono dai cicli appartenenti a $\mathscr{N}$ sono le seguenti:

$$u_2 \wedge_c \delta_4^2 = 0\,; \qquad u_2 \wedge_c \delta_4^3 = 0\,; \qquad u_3 \wedge_c \delta_4^3 = 0\,;$$

$$u_4 \wedge_c \delta_4^1 = 0\,; \qquad u_4 \wedge_c \delta_4^3 = 0\,; \qquad u_4 \wedge_c \delta_4^4 = 0\,;$$

$$(u_1 \wedge_c \delta_4^3) \wedge (u_4 \wedge_c \delta_4^1) = 0\,; \quad (u_2 \wedge_c \delta_4^4) \wedge (u_3 \wedge_c \delta_4^2) = 0\,.$$

Le equazioni ottenute sono 8 perchè i cicli $\mathscr{C}_2$ e $\mathscr{C}_3$ sono relativi a stati di T_2 e entrambi impongono che l'ingresso relativo a u_2 sia diverso da δ_4^3. Gli ingressi che rispettano queste equazioni e che sono tali per cui x_e è punto di equilibrio, e le relative matrici di retroazione dall'uscita, sono le seguenti:

$$\begin{aligned}
&\mathbf{1}: u_1 = 1;\ u_2 = 1;\ u_3 = 4;\ u_4 = 3; \quad \Rightarrow \quad K_1 = \delta_4 \left[\begin{array}{cccc} 1 & 1 & 4 & 3 \end{array}\right];\\
&\mathbf{2}: u_1 = 1;\ u_2 = 1;\ u_3 = 2;\ u_4 = 3; \quad \Rightarrow \quad K_2 = \delta_4 \left[\begin{array}{cccc} 1 & 1 & 2 & 3 \end{array}\right];\\
&\mathbf{3}: u_1 = 1;\ u_2 = 1;\ u_3 = 1;\ u_4 = 3; \quad \Rightarrow \quad K_3 = \delta_4 \left[\begin{array}{cccc} 1 & 1 & 1 & 3 \end{array}\right];\\
&\mathbf{4}: u_1 = 1;\ u_2 = 4;\ u_3 = 4;\ u_4 = 3; \quad \Rightarrow \quad K_4 = \delta_4 \left[\begin{array}{cccc} 1 & 4 & 4 & 3 \end{array}\right];\\
&\mathbf{5}: u_1 = 1;\ u_2 = 4;\ u_3 = 1;\ u_4 = 3; \quad \Rightarrow \quad K_5 = \delta_4 \left[\begin{array}{cccc} 1 & 4 & 1 & 3 \end{array}\right];\\
&\mathbf{6}: u_1 = 3;\ u_2 = 1;\ u_3 = 2;\ u_4 = 3; \quad \Rightarrow \quad K_6 = \delta_4 \left[\begin{array}{cccc} 3 & 1 & 2 & 3 \end{array}\right];\\
&\mathbf{7}: u_1 = 3;\ u_2 = 1;\ u_3 = 1;\ u_4 = 3; \quad \Rightarrow \quad K_7 = \delta_4 \left[\begin{array}{cccc} 3 & 1 & 1 & 3 \end{array}\right];\\
&\mathbf{8}: u_1 = 3;\ u_2 = 4;\ u_3 = 1;\ u_4 = 3; \quad \Rightarrow \quad K_8 = \delta_4 \left[\begin{array}{cccc} 3 & 4 & 1 & 3 \end{array}\right];\\
&\mathbf{9}: u_1 = 3;\ u_2 = 1;\ u_3 = 4;\ u_4 = 3; \quad \Rightarrow \quad K_9 = \delta_4 \left[\begin{array}{cccc} 3 & 1 & 4 & 3 \end{array}\right];\\
&\mathbf{10}: u_1 = 3;\ u_2 = 4;\ u_3 = 4;\ u_4 = 3; \quad \Rightarrow \quad K_{10} = \delta_4 \left[\begin{array}{cccc} 3 & 4 & 4 & 3 \end{array}\right].
\end{aligned}$$

I risultati ottenuti sono identici a quelli ottenuti con il metodo della matrice polinomiale, come era naturale aspettarsi. ♣

Questi due approci esposti sono in grado di pervenire alla matrice K di retroazione dall'uscita statica tempo invariante studiando la BCN e le

sue caratteristiche.
Un altro approcio per raggiungere lo stesso scopo è basato sul metodo a forza bruta; la tecnica consiste cioè nell'enumerare tutte le possibili matrici di retroazione e di controllarle una per una fino a quando non se ne trovi una che permette la retroazione stabilizzante a x_e. Questa tecnica risulta alquanto onerosa quando il numero degli ingressi e delle uscite è molto elevato, dal momento che le possibili matrici di retroazione sono in numero $(2^m)^{2^p}$, quindi all'aumentare degli ingressi o delle uscite (o di entrambi), le matrici da testare diventano molto numerose; inoltre questo metodo presenta lo svantaggio che se la retroazione non esiste, questo può solo essere stabilito dopo aver controllato che nessuna delle possibili matrici di retroazione svolga il compito desiderato.
Esistono diversi modi per implementare questa tecnica a forza bruta, il primo e più semplice è quello di calcolare la matrice $\tilde{L}$ per ogni scelta della matrice K e di controllare che la sua potenza 2^n-esima presenti tutte le colonne pari a $\delta^1_{2^n}$, come è già stato sottolineato all'inizio del paragrafo.
Un altro metodo è quello di trovare la forma normale della matrice $\tilde{L}$ e di controllare che questa presenti una sola classe ergodica di dimensione 1 che contiene lo stato $x_e = \delta^1_{2^n}$.
Queste due soluzioni non sono state implementate in Matlab, e si è preferito implementarne un'altra che applica un ragionamento simile ma più sofisticato, che viene di seguito spiegato.
Tutti i metodi appena illustrati non sfruttano minimamente la conoscenza della rete, ed ogni matrice di retroazione K viene testata per stabilire se è di retroazione o meno; è in realtà possibile adottare una tecnica più elaborata che scarta alcune di queste matrici perchè risultano certamente non compatibili con l'obiettivo prefissato. Per sua natura, la matrice $\tilde{L}$, che si ricava prendendo i blocchi colonna della matrice L relativi alla retroazione scelta, è una matrice positiva; inoltre, se la retroazione svolge l'obbiettivo richiesto, la potenza 2^n-esima di $\tilde{L}$ ha tutte le colonne pari a $\delta^1_{2^n}$.
Se si considera una matrice quadrata positiva $A \in \mathbb{R}^n_+$, e si considera una sua sottomatrice principale A_i ottenuta dalle righe/colonne consecutive $i_1, \ldots, i_k,\ k < N$, è facile convincersi che condizione necessaria affinchè A^n presenti come sottomatrice principale relativa agli indici $i_1, \ldots, i_k$ una matrice nulla di dimensione k, è che la matrice A_i sia una matrice nilpotente. Questa è una evidente conseguenza della positività della matrice A, e del fatto che A_i è una sua sottomatrice principale.
Grazie a questa considerazione è chiaro che considerando l'insieme $\mathcal{T}_i, 2 \leq i \leq 2^p$, formato dagli stati consecutivi $\delta^{i_1}_{2^n}, \ldots, \delta^{i_k}_{2^n},\ k = |\mathcal{T}_i|$, che forniscono come uscita $\delta^i_{2^p}$, possono essere usati come ingressi $\delta^j_{2^m}$ relativi a

questa uscita, solo quelli per cui la matrice $L_{j,i}$ presenta la sottomatrice ricavata prendendo tutte le colonne e le righe di indice $i_1, \ldots, i_k$ nilpotente. Questo è dovuto al fatto che in $\tilde{L}$ la sottomatrice appena ricavata è una sottomatrice principale.
Per quanto riguarda gli ingressi relativi all'uscita $\delta^1_{2^p}$, l'ingresso $\delta^j_{2^m}$ è accettabile se in posizione $(1,1)$ la matrice $L_{j,1}$ presenta un 1 (perchè $\delta^1_{2^n}$ deve essere stato di equilibrio), e se la sua sottomatrice ricavata prendendo tutte le colonne tranne la prima, e prendendo le righe di indice $2, \ldots, k,\ k = |\mathcal{T}_1|$, risulta nilpotente (nel caso lo stato $\delta^1_{2^n}$ sia l'unico a fornire come uscita $\delta^1_{2^p}$, l'unica condizione da controllare è la prima).
Fatta una prima selezione dei possibili ingressi, si procede poi componendo la matrice $\tilde{L}$ per passi successivi, cioè si inseriscono prima i blocchi $L_{j_1,1}, L_{j_2,1}$, prendendo j_1 e j_2 dagli ingressi possibili per $\mathcal{T}_1$ e $\mathcal{T}_2$ e in modo che la sottomatrice $[\tilde{L}]_{1:2}$ formata da tutte le colonne tranne la prima e dalle righe $2, \ldots, |\mathcal{T}_1 \cup \mathcal{T}_2|$ sia nilpotente. A questo punto si aggiunge il blocco colonna relativo a $\mathcal{T}_3$, cercando un ingresso per questo in modo che la sottomatrice $[\tilde{L}]_{1:3}$ ricavata da tutte le colonne tranne la prima e dalle righe $2, \ldots, |\mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{T}_3|$ sia nilpotente; se non si dovesse trovare, bisogna cercare un'altra scelta degli ingressi per $\mathcal{T}_1$ e $\mathcal{T}_2$ tale che $[\tilde{L}]_{1:2}$ sia nilpotente e per la quale esista un ingresso per $\mathcal{T}_3$ per cui $[\tilde{L}]_{1:3}$ sia nilpontente. Nel caso non si riuscisse a trovare nessuna combinazione di ingressi per $\mathcal{T}_1, \mathcal{T}_2$ e $\mathcal{T}_3$, si può concludere che non esiste la retroazione dall'uscita tempo invariante desiderata. Se si trovano combinazioni di ingresso che rispettano le condizoni richieste fino a $\mathcal{T}_{2^p}$, si ottiene che la matrice $\tilde{L}$, ottenuta con questi ingressi, elevata alla 2^n-esima potenza ha tutte le colonne pari a $\delta^i_{2^n}$. Questo è dovuto al fatto che la sottomatrice $[\tilde{L}]_{1:2^p}$ è nilpotente, e la prima colonna di $\tilde{L}$ ha solo il primo elemento diverso da 0, e resta tale per ogni potenza di $\tilde{L}$. Quindi, per $2 \leq i, j \leq 2^n$, si ha

$$[\tilde{L}^2]_{ij} = [\tilde{L}]_{i1}[\tilde{L}]_{1j} + \sum_{k=2}^{N} [\tilde{L}]_{i_k}[\tilde{L}]_{k_j} = \sum_{k=2}^{N} [\tilde{L}]_{i_k}[\tilde{L}]_{k_j} \ .$$

Essendo valido per ogni elemento di $[\tilde{L}]_{1:2^p}$, si ha $[\tilde{L}^2]_{1:2^p} = [\tilde{L}]^2_{1:2^p}$, e questo ragionamento può essere ripetuto fino ad ottenere che $[\tilde{L}^{2^n}]_{1:2^p} = [\tilde{L}]^{2^n}_{1:2^p}$, ma poichè la matrice in questione è nilpotente allora il tutto è necessariamente uguale a 0, mentre la prima riga di $\tilde{L}^{2^n}$ deve essere composta da tutti 1, dal momento che $\tilde{L}$ è stocastica per colonne, e ogni colonna presenta un solo elemento diverso da 0 e quindi pari a 1, e questo resta valido per una sua potenza qualsiasi.

Esempio 4.8: Implementando in Matlab quest'ultimo metodo esposto, si ottengono, per le BCN degli Esempi 4.4 e 4.2, le stesse matrici di retroazione ricavate con i due metodi precedenti, a conferma che il metodo illustrato ha la stessa efficacia degli altri due nel ritrovare la retroazione cercata, pur applicando una tecnica diversa. ♣

4.2.3 Retroazione tempo variante

Un altro tipo di retroazione dall'uscita che si può effettuare è di tipo statico ma tempo variante: l'ingresso cioè risulta sempre funzione dell'uscita del sistema al tempo t, ma la matrice di retroazione K non risulta costante nel tempo, cambiando in funzione dell'istante temporale, per cui $K = K(t)$ e $u(t) = K(t)y(t)$. Questo tipo di retroazione ovviamente è molto più generale della retroazione trattata precedentemente, e permetterà di risolvere il problema di stabilizzazione della rete anche in molti casi in cui non esiste la retroazione tempo invariante; ciononostante esistono delle reti in cui anche questo tipo di retroazione dall'uscita non è in grado di risolvere il problema di stabilizzazione.
Come ipotesi iniziale si suppone che la rete sia stabilizzabile allo stato x_e, e per comodità si utilizzano le stesse ipotesi introdotte all'inizio del Paragrafo 4.2.2, riguardanti la numerazione degli stati e delle uscite della BCN. Riprendendo la notazione introdotta nei paragrafi precedenti, considerato uno stato $x = \delta_{2^n}^i$, $2 \leq i \leq 2^n$, appartenente alla corona $\mathcal{S}_k$, $1 \leq k \leq 2^n - 1$, con P_i si indica l'insieme degli ingressi che permettono allo stato x di giungere in un passo ad uno stato appartenente alla corona $\mathcal{S}_{k-1}$, mentre nel caso x appartenga a $\mathcal{S}_0$, cioè sia lo stato di equilibrio, in P_1 sono contenuti gli ingressi che mantengono lo stato in se stesso. Inoltre, l'insieme $\mathcal{T}_j$, $1 \leq j \leq 2^p$, contiene tutti gli stati che forniscono come uscita $\delta_{2^p}^j$.
Una prima considerazione riguarda tutti gli stati appartenenti a $\mathcal{T}_1$: questi possono ricevere solo ingressi che sono di equilibrio per x_e, e quindi solo ingressi che appartengono a P_1, per assicurare che, se lo stato raggiunto al tempo t dalla BCN è x_e, la rete non si sposti dallo stesso. Se la seguente ipotesi è verificata, si dimostra che la retroazione dall'uscita tempo variante esiste:

- x_e deve essere raggiungibile da ogni stato della rete grazie all'uso di ingressi che, per gli stati appartenenti a $\mathcal{T}_1$, sono di equilibrio per x_e, e quindi appartengono a P_1.

Se una BCN è identificata dalle matrici L e H, per verificare che questa rete rispetti l'ipotesi appena esposta, è possibile creare un'altra rete Booleana, che nel seguito viene indicata soprasegnando le matrici che la identificano, avente $\bar{H} = H$ e la matrice $\bar{L}$ che coincide con L per tutti gli stati che non restituiscono $\delta_{2^p}^1$ come uscita e per lo stato di equilibrio; per un qualsiasi altro stato $\delta_{2^n}^j$, $2 \leq j \leq 2^n$, appartenente a $\mathcal{T}_1$ e diverso dallo stato di equilibrio la matrice $\bar{L}$ della nuova rete coincide con la prima per gli ingressi che appartengono a P_1, mentre presenta le colonne relative agli altri ingressi pari a $\delta_{2^n}^j$. In questo modo per gli ingressi non di equilibrio per $\delta_{2^n}^1$, gli stati di $\mathcal{T}_1$ sono punti fissi, quindi se nella rete determinata da $\bar{L}$, lo stato x_e risulta raggiungibile a partire da ogni stato, l'ipotesi è assicurata. Infatti, poiché nella nuova rete gli ingressi non di equilibrio per x_e non possono condurre alcun stato di $\mathcal{T}_1$ verso x_e, essendo di equilibrio per quegli stessi stati, se x_e è raggiungibile da questi stati, lo è attraverso ingressi che sono di equilibrio per x_e stesso.
A questo punto si introduce l'algoritmo con cui si può ricavare la sequenza di matrici di retroazione grazie alla quale è possibile stabilizzare il sistema a x_e. L'algoritmo pensato per trovare la sequenza delle matrici di retroazione che risolvono il problema non è in tempo reale, per cui nel suo svolgimento non viene sfruttata la conoscenza dell'uscita in quell'istante, ma si crea una sequenza di matrici $K(\cdot)$ che risolverà il problema per qualunque evoluzione del sistema.
La suddivisione in corone $\mathcal{S}_k$, $k = 0, \ldots, \bar{k}$, $1 \leq \bar{k} \leq 2^n - 1$, e gli ingressi P_i, $i = 1, \ldots, 2^n$, per il passaggio da una corona a quella inferiore che vengono usati nell'algoritmo, non corrispondono agli insiemi corrispondenti nella rete di partenza, ma sono quelli che si ricavano dalla rete modificata, in modo da assicurare l'uso di ingressi di equilibrio per x_e ogni qualvolta l'uscita del sistema sia $\delta_{2^p}^1$.
Prendendo 0 come istante di inizio del controllo, lo stato in cui può trovarsi la rete risulta totalmente sconosciuto all'algoritmo; le operazioni compiute da quest'ultimo sono le seguenti:

- operazioni da svolgere al tempo t=0:
  - se l'insieme $\mathcal{T}_1$ ha cardinalità maggiore di 1, si trova il più piccolo intero $\tilde{k}$ per cui $\mathcal{T}_1 \cap \mathcal{S}_{\tilde{k}} \neq \varnothing$; si sceglie quindi uno degli stati che appartengono a questo insieme, $x = \delta_{2^n}^i$, e l'ingresso al tempo 0 corrispondente all'uscita $\delta_{2^p}^1$ viene preso nell'insieme P_i;
    se $\mathcal{T}_1$ è composto solo dallo stato x_e, si sceglie come ingresso per l'uscita $\delta_{2^p}^1$ uno degli elementi appartenenti a P_0;

- per ogni insieme $\mathcal{T}_j$, $2 \leq j \leq 2^p$, si trova il più piccolo intero $\tilde{k}$ per cui $\mathcal{T}_j \cap \mathcal{S}_{\tilde{k}} \neq \varnothing$; si sceglie quindi uno degli stati che appartengono a questo insieme, $x = \delta^i_{2^n}$, e l'ingresso al tempo 0 corrispondente all'uscita $\delta^j_{2^p}$ viene preso nell'insieme P_i (se l'insime $\mathcal{T}_j$ dovesse essere vuoto si può scegliere un ingresso qualsiasi);
- a questo punto vengono creati gli insiemi $\mathcal{T}_\ell(1), 1 \leq \ell \leq 2^p$: per ogni diverso $\mathcal{T}_k$, $1 \leq k \leq 2^p$, si prende ogni suo elemento $x_p = \delta^i_{2^n}$ e si calcola lo stato $x_a = \delta^j_{2^n}$ di arrivo grazie all'ingresso deciso per gli stati di $\mathcal{T}_k$; si calcola l'uscita $\delta^h_{2^p}$ generata dallo stato x_a e quindi si inserisce x_a in $\mathcal{T}_h(1)$;

- per ogni istante $t > 0$ si ripetono le 3 operazioni effettuate ai punti precedenti, considerando al posto di $\mathcal{T}_i$ gli insiemi $\mathcal{T}_i(t)$, e creando all'ultimo punto gli insiemi $\mathcal{T}_i(t+1)$. Queste operazioni vanno effettuate finchè $\mathcal{T}_1(t+1) = \{\delta^1_{2^n}\}$ e $\mathcal{T}_i(t+1) = \varnothing, 2 \leq i \leq 2^n$.

Ovviamente per costruzione gli insiemi $\mathcal{T}_\ell(t), 1 \leq \ell \leq 2^p$, contengono gli stati, che restituiscono come uscita $\delta^\ell_{2^p}$, in cui la rete può essere all'istante t.
In Algoritmo 4.1 è presente lo pseudocodice dell'algoritmo appena presentato.

Teorema 4.5 *Se x_e è raggiungibile da ogni stato della rete grazie ad ingressi che per gli stati appartenenti a $\mathcal{T}_1$ sono di equilibrio per x_e, allora la BCN (3.3) è stabilizzabile grazie alla retroazione dall'uscita utilizzando la sequenza di matrici di retroazione $K(t)$ ricavate con l'Algoritmo 4.1.*

Dimostrazione: La sequenza di matrici di retroazione trovata grazie all'algoritmo assicura che ad ogni istante ci sia almeno uno stato che passa in una corona inferiore. Ad ogni istante t infatti, in ogni settore $\mathcal{T}_i(t)$ diverso dall'insieme nullo, l'ingresso scelto fa in modo che almeno uno degli stati appartenenti alla corona più bassa (fra tutte le corone degli stati di $\mathcal{T}_i(t)$) scenda ulteriormente di corona; questo fatto comporta che all'istante $t+1$ in ogni settore (che al passo precedente non era vuoto), la corona più bassa che ha almeno uno stato in $\mathcal{T}_i(t+1)$ ha un indice inferiore rispetto alla corona più bassa dell'istante precedente e quindi reiterando l'operazione, ad un certo istante $\bar{t}$ esiste certamente un $\mathcal{T}_i(\bar{t})$ contenente almeno uno stato che appartiene alla corona $\mathcal{S}_1$; al passo successivo l'algoritmo porterà almeno uno di questi stati in x_e (dal momento che la corona più bassa è proprio $\mathcal{S}_1$) e, poiché gli ingressi scelti per $\mathcal{T}_1(t)$ sono sempre di equilibrio per x_e (grazie all'ipotesi di

Algoritmo 4.1 Algoritmo per la costruzione delle matrici di retroazione dall'uscita tempo variante.

Input: P_i, $1 \leq i \leq 2^n$; $\mathcal{T}_j$, $1 \leq j \leq 2^p$; $\mathcal{S}_k$, $k = 0, \ldots, \bar{k}$;
Output: $K(\cdot)$, sequenza delle matrici di retroazione;

$t = 0$;
while $\mathcal{T}_1 \neq \{\delta_{2^n}^1\} \ \wedge \ \mathcal{T}_j \neq \varnothing$, $2 \leq j \leq 2^p$ **do**
  if $|\mathcal{T}_1| > 1$ **then**
    $\tilde{k}$=minimo k tale per cui $\mathcal{T}_1 \cap \mathcal{S}_k \neq \varnothing$;
    $stInf$=stato di $\mathcal{T}_1 \cap \mathcal{S}_{\tilde{k}}$;
    ing= ingresso appartenente a P_{stInf};
    $K(t) = [ing]$;
  else
    ing= ingresso appartenente a P_1;
    $K(t) = [ing]$;
  end if
  for $i = 2, \ldots, 2^p$ **do**
    if $|\mathcal{T}_i| > 0$ **then**
      $\tilde{k}$=minimo k tale per cui $\mathcal{T}_i \cap \mathcal{S}_k \neq \varnothing$;
      $stInf$=stato di $\mathcal{T}_i \cap \mathcal{S}_{\tilde{k}}$;
      ing= ingresso appartenente a P_{stinf};
      $K(t) = [K(t)\ ing]$;
    else
      ing= ingresso qualunque;
      $K(t) = [K(t)\ ing]$;
    end if
  end for
  $\mathcal{T}_i^n$=insiemi $\mathcal{T}_i$ aggiornati;
  for $i = 1, \ldots, 2^p$ **do**
    if $|\mathcal{T}_i| > 0$ **then**
      for ogni elemento x di $\mathcal{T}_i$ **do**
        x_a=stato di arrivo partendo da x con ingresso $[K(t)]_i$;
        $\delta_{2^p}^h$=uscita generata da stato x_a;
        $\mathcal{T}_h^n = \mathcal{T}_h^n \cup \{x_a\}$;
      end for
    end if
  end for
  for $i = 1, \ldots, 2^p$ **do**
    $\mathcal{T}_i = \mathcal{T}_i^n$;
  end for
  $t++$;
end while

raggiungibilità), gli stati non ancora assorbiti da x_e sono diminuiti di almeno una unità, e non potranno aumentare successivamente. Dopo un numero finito di passi si ottiene quindi la stabilizzazione allo stato x_e.
$\diamondsuit$

Esempio 4.9: Si riprende nuovamente la BCN dell'Esempio 4.2, con l'obbiettivo di stabilizzare il sistema a $x_e = \delta_{2^n}^1$ attraverso retroazione dall'uscita tempo variante; la suddivisione degli stati nelle corone, gli ingressi da dare in corrispondenza di ogni stato per arrivare in uno stato appartenente a una corona inferiore sono i seguenti (già riportati nell'Esempio citato) e la suddivisione nei diversi $\mathcal{T}_i$ sono:

$$\mathcal{S}_0 = \{1\};\ \mathcal{S}_1 = \{7,4\};\ \mathcal{S}_2 = \{2,5\};\ \mathcal{S}_3 = \{6,8\};\ \mathcal{S}_4 = \{3\}\ ;$$

$$P_1 = \{1,3\};\ P_2 = \{1,4\};\ P_3 = \{1,4\};\ P_4 = \{1,2\};\ P_5 = \Delta_4;$$
$$P_6 = \{2,4\};\ P_7 = \{2,3,4\};\ P_8 = \{1,3\}\ ;$$
$$\mathcal{T}_1 = \{1,2\};\ \mathcal{T}_2 = \{3,4\};\ \mathcal{T}_3 = \{5,6\};\ \mathcal{T}_4 = \{7,8\}\ .$$

Dal momento che $P_1 \cap P_2 = \{1\}$, se per lo stato 2 si sceglie come ingresso che porta alla corona inferiore solo l'ingresso 1, e quindi $P_2 = \{1\}$, l'ipotesi è verificata (non è stato necessario calcolare la rete Booleana modificata perchè già in quella di partenza gli ingressi di equilibrio assicuravano la raggiungibilità di x_e da parte dello stato 2, ed è stato sufficiente non considerare l'altro ingresso). Si riportano quindi i diversi $\mathcal{T}_i$ nel tempo e le matrici $K(t)$ che possono essere usate per la retroazione (nel caso ci fossero più possibili scelte si prende l'elemento di P_i minore).

$$K(0) = \left[\begin{array}{cccc} 1 & 1 & 1 & 2 \end{array}\right] \Leftrightarrow \begin{array}{l} \mathcal{T}_1(1) = \{1\};\ \mathcal{T}_2(1) = \varnothing;\ \mathcal{T}_3(1) = \{6\}; \\ \mathcal{T}_4(1) = \{7,8\}. \end{array}$$

$$K(1) = \left[\begin{array}{cccc} 1 & 1 & 2 & 2 \end{array}\right] \Leftrightarrow \begin{array}{l} \mathcal{T}_1(2) = \{1\};\ \mathcal{T}_2(2) = \varnothing;\ \mathcal{T}_3(2) = \{5\}; \\ \mathcal{T}_4(2) = \{8\}. \end{array}$$

$$K(2) = \left[\begin{array}{cccc} 1 & 1 & 1 & 1 \end{array}\right] \Leftrightarrow \begin{array}{l} \mathcal{T}_1(3) = \{1,2\};\ \mathcal{T}_2(3) = \varnothing;\ \mathcal{T}_3(3) = \varnothing; \\ \mathcal{T}_4(3) = \{7\}. \end{array}$$

$$K(3) = \left[\begin{array}{cccc} 1 & 1 & 1 & 2 \end{array}\right] \Leftrightarrow \begin{array}{l} \mathcal{T}_1(4) = \{1\};\ \mathcal{T}_2(4) = \varnothing;\ \mathcal{T}_3(4) = \varnothing; \\ \mathcal{T}_4(4) = \{7\}. \end{array}$$

$$K(4) = \left[\begin{array}{cccc} 1 & 1 & 1 & 2 \end{array}\right] \Leftrightarrow \begin{array}{l} \mathcal{T}_1(5) = \{1\};\ \mathcal{T}_2(5) = \varnothing;\ \mathcal{T}_3(5) = \varnothing; \\ \mathcal{T}_4(5) = \varnothing. \end{array}$$

È stata quindi individuata una soluzione tempo variante per lo stesso problema posto negli Esempi precedenti; si sottolinea che per esempio applicando la prima matrice di retroazione tempo invariante, il numero di passi necessari per stabilizzare il sistema a x_e è 4, mentre con quest'ultimo metodo la stabilizzazione avviene in 5 passi, quindi non è detto ci siano dei miglioramenti rispetto al tempo di stabilizzazione. Con il prossimo esempio si mostra che il campo di applicabilità della retroazione tempo variante è più ampio rispetto a quello della retroazione tempo invariante.
♣

Esempio 4.10: Si consideri la seguente BCN, che presenta 16 stati, 4 ingressi e 4 uscite, in cui la matrice L è fornita attraverso i 4 sottosistemi che la compongono:

$$L_1 = \delta_{16}\left[\begin{array}{cccccccccccccccc} 1 & 2 & 1 & 6 & 4 & 9 & 7 & 4 & 8 & 10 & 9 & 14 & 12 & 15 & 5 & 14 \end{array}\right]$$

$$L_2 = \delta_{16}\left[\begin{array}{cccccccccccccccc} 10 & 1 & 7 & 13 & 4 & 6 & 4 & 7 & 6 & 5 & 1 & 12 & 2 & 11 & 12 & 10 \end{array}\right]$$

$$L_3 = \delta_{16}\left[\begin{array}{cccccccccccccccc} 14 & 3 & 3 & 5 & 5 & 9 & 3 & 6 & 10 & 12 & 1 & 13 & 2 & 14 & 9 & 8 \end{array}\right]$$

$$L_4 = \delta_{16}\left[\begin{array}{cccccccccccccccc} 1 & 13 & 2 & 13 & 15 & 4 & 1 & 8 & 9 & 9 & 7 & 14 & 13 & 10 & 12 & 16 \end{array}\right]$$

$$H = \delta_4\left[\begin{array}{cccccccccccccccc} 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 3 & 3 & 3 & 4 & 4 & 4 & 4 & 4 \end{array}\right]$$

La suddivisione degli stati a seconda dell'uscita che producono è la seguente:

$\mathcal{T}_1 = \{1,2,3,4,5\}$; $\mathcal{T}_2 = \{6,7,8\}$; $\mathcal{T}_3 = \{9,10,11\}$; $\mathcal{T}_4 = \{12,13,14,15,16\}$.

L'obiettivo richiesto è quello di stabilizzare la rete allo stato $x_e = \delta_{2^n}^1$ attraverso una retroazione dall'uscita.
Se si volesse effettuare una retroazione dall'uscita tempo invariante, nel settore $\mathcal{T}_1$ l'unico ingresso utilizzabile sarebbe il quarto perchè gli ingressi 1 e 3 sono di equilibrio rispettivamente per gli stati 2 e 3, mentre l'ingresso 2 non è di equilibrio per x_e. Per quanto riguarda il settore $\mathcal{T}_4$, gli ingressi 2,3 e 4 sono di equilibrio per almeno uno stato di $\mathcal{T}_4$, e quindi l'unico ingresso che potrebbe essere applicato è l'ingresso 1. Con questi 2 ingressi per il settore $\mathcal{T}_1$ e $\mathcal{T}_4$ però si attiva un ciclo che coinvolge gli stati 5 e 15, e quindi risulta impossibile effettuare retroazione dall'uscita tempo invariante.
La tecnica tempo variante però risulta efficace in questa rete; a differenza dell'Esempio precedente, è qui necessario introdurre una BCN modificata perché lo stato 2 potrebbe in un passo raggiungere lo stato di equilibrio, ma solo attraverso un ingresso non appartenente a P_1. Con la

rete ausiliaria è immediato verificare il soddisfacimento dell'ipotesi sulla raggiungibilità ed è facile ottenere la seguente suddivisione degli stati in corone e gli ingressi relativi agli stati per passare alla corona inferiore:

$$\mathcal{S}_0 = \{1\};\ \mathcal{S}_1 = \{3,7,11\};\ \mathcal{S}_2 = \{8,14\};\ \mathcal{S}_3 = \{9,12,16\};$$

$$\mathcal{S}_4 = \{6,10,13,14\};\ \mathcal{S}_5 = \{2,4,5\};$$

$$P_1 = \{1,4\};\ P_2 = \{4\};\ P_3 = \{1\};\ P_4 = \{1,4\};\ P_5 = \{4\};\ P_6 = \{1,3\};$$

$$P_7 = \{4\};\ P_8 = \{2\};\ P_9 = \{1\};\ P_{10} = \{3,4\};\ P_{11} = \{2,3\};\ P_{12} = \{1,4\};$$

$$P_{13} = \{1\};\ P_{14} = \{2\};\ P_{15} = \{2,3,4\};\ P_{16} = \{1,3\}.$$

Si evidenzia il fatto che lo stato 2, che potrebbe raggiungere lo stato di equilibrio in un passo e quindi apparterrebbe alla corona $\mathcal{S}_1$ se non ci fossero ulteriori vincoli, non potendo sfruttare l'ingresso che permette questo passaggio perché non di equilibrio per x_e, appartiene alla corona $\mathcal{S}_5$.
Si può facilmente verificare che in 11 passi, grazie alla seguente sequenza di matrici di retroazione, la rete è stata stabilizzata allo stato $x_e = \delta_{2^n}^1$:

$$K(0) = \delta_4 \left[\begin{array}{cccc} 1 & 4 & 2 & 2 \end{array}\right]; \qquad K(1) = \delta_4 \left[\begin{array}{cccc} 4 & 2 & 2 & 1 \end{array}\right];$$

$$K(2) = \delta_4 \left[\begin{array}{cccc} 4 & 4 & 1 & 2 \end{array}\right]; \qquad K(3) = \delta_4 \left[\begin{array}{cccc} 1 & 1 & 2 & 1 \end{array}\right];$$

$$K(4) = \delta_4 \left[\begin{array}{cccc} 4 & 1 & 1 & 2 \end{array}\right]; \qquad K(5) = \delta_4 \left[\begin{array}{cccc} 1 & 1 & 2 & 1 \end{array}\right];$$

$$K(6) = \delta_4 \left[\begin{array}{cccc} 4 & 1 & 1 & 1 \end{array}\right]; \qquad K(7) = \delta_4 \left[\begin{array}{cccc} 1 & 1 & 1 & 2 \end{array}\right];$$

$$K(8) = \delta_4 \left[\begin{array}{cccc} 1 & 2 & 2 & 1 \end{array}\right]; \qquad K(9) = \delta_4 \left[\begin{array}{cccc} 1 & 4 & 1 & 2 \end{array}\right];$$

$$K(10) = \delta_4 \left[\begin{array}{cccc} 1 & 1 & 2 & 1 \end{array}\right].$$

Risulta quindi evidente che con la retroazione dall'uscita tempo variante è possibile stabilizzare ad uno stato di equilibrio un numero maggiore di BCN rispetto al numero di reti stabilizzabili con una retroazione dall'uscita tempo invariante. ♣

La condizione dettata dal Teorema 4.5 è solo sufficiente, per cui se in una rete x_e non è raggiungibile da tutti gli stati con ingressi che per gli stati di $\mathcal{T}_1$ sono di equilibrio per x_e, non è detto che non esista comunque una retroazione tempo variante in grado di stabilizzare la BCN allo stato desiderato.

4.2.4 Retroazione tramite ricostruzione dello stato

In questo breve paragrafo si accenna infine ad un'ulteriore soluzione a livello teorico, senza entrare nei dettagli. La soluzione che viene proposta in questo contesto necessita dell'ipotesi di ricostruibilità della rete, ipotesi grazie alla quale si è in grado di determinare, ad un certo istante temporale $\bar{t}$, quale sia lo stato $x(\bar{t})$ conoscendo simultaneamente ingressi e uscite della BCN (si confronti la Definizione 3.9). Assumendo l'indispensabile ipotesi di stabilizzabilità della BCN, si può enunciare il seguente

Teorema 4.6 *Se la BCN (3.3) è stabilizzabile allo stato x_e ed è ricostruibile, allora esiste una retroazione dall'uscita in grado di stabilizzare il sistema allo stato x_e.*

Dimostrazione: Grazie all'ipotesi di ricostruibilità, da una qualunque sequenza di ingresso e dalla conseguente sequenza di uscita, ad un certo istante $\bar{t} < \infty$ si è in grado di stabilire lo stato $x(\bar{t})$ raggiunto dalla rete realizzando una rete Booleana, che agisce come ricostruttore deadbeat dello stato [3]. In questo modo dall'istante $\bar{t}$ in poi, pur misurando direttamente solo l'uscita e l'ingresso, si è in grado di conoscere lo stato vero e proprio. Grazie all'ipotesi di stabilizzabilità vale la Proposizione 4.1, quindi è possibile dall'istante $\bar{t}$ in poi applicare una retroazione dallo stato, la cui esistenza è assicurata, ottenendo il risultato desiderato. $\diamondsuit$

Osservazione 4.9: Nel caso la BCN in analisi sia ricostruibile, questo tipo di retroazione dall'uscita è in grado di dare una soluzione al problema. Tuttavia la condizione di ricostruibilità non è necessaria affinché si possa stabilizzare il sistema attraverso una retroazione dall'uscita. Se per esempio si analizza la rete presente nell'Esempio 4.3, si verifica immediatamente che la rete non è ricostruibile. Secondo il Teorema 3.6 infatti, la BCN in esame non è ricostruibile a causa dell'esistenza dei cicli

$$\left((\delta_4^1, \delta_2^2), (\delta_4^4, \delta_2^2)\right) \neq \left((\delta_4^4, \delta_2^2), (\delta_4^1, \delta_2^2)\right) \ ,$$

dal momento che

$$H\delta_4^1 = H\delta_4^4 \ .$$

Per la rete in esame non è quindi possibile applicare il metodo appena esposto; ciononostante con l'Esempio 4.4 si è dimostrato che è possibile, attraverso una retroazione tempo invariante dall'uscita, stabilizzare il sistema ad x_e. $\square$

4.3 Controllo ottimo

L'ultima strategia di controllo di una BCN che viene riportata è il controllo ottimo; numerosi aspetti di questo problema sono stati trattati in letteratura, per esempio in [8], [9], [10], [1]. La trattazione qui seguita si basa su [7], dove sono tenuti in considerazione anche gli aspetti studiati nei lavori sopra citati.
Lo scopo del controllo ottimo è quello di individuare una sequenza di controllo per la BCN che minimizzi (massimizzi) una data funzione di costo (di qualità). Lo studio può essere effettuato ad orizzonte finito, cioè in un numero finito di passi, oppure ad orizzonte infinito, cioè studiando una sequenza di ingresso di lunghezza infinita; inoltre la funzione di costo può pesare aspetti diversi, quali: lo stato finale, l'evoluzione dello stato e dell'ingresso o il cambiamento dei valori dell'ingresso, etc.
Inizialmente verrà trattato il controllo ottimo ad orizzonte finito e successivamente quello ad orizzonte infinito.

4.3.1 Orizzonte finito

Definizione 4.2: Data una BCN (3.3) con stato iniziale $x(0) = x_0 \in \Delta_{2^n}$ e fissato un istante $T > 0$, il problema del *controllo ottimo ad orizzonte finito* consiste nel determinare una sequenza finita di ingresso che minimizza il funzionale quadratico di costo

$$J_T(x_0, u(\cdot)) = x(T)^T Q_f x(T) + \sum_{t=0}^{T-1} x(t)^T Q_{u(t)} x(t) \,, \tag{4.3}$$

dove $Q_f = Q_f^T \in \mathbb{R}^{2^n \times 2^n}$ è una matrice semidefinita positiva che pesa lo stato finale, mentre $Q_{u(t)} = Q_{u(t)}^T$ è una matrice semidefinita positiva che per ogni valore di $u(t) \in \Delta_{2^m}$ assume valori in $\mathbb{R}^{2^n \times 2^n}$, e pesa ad ogni istante intermedio lo stato $x(t)$ raggiunto.

Per semplicità, se all'istante t l'ingresso vale $u(t) = \delta_{2^m}^i$, la conseguente matrice di costo $Q_{u(t)} = Q_{\delta_{2^m}^i}$ verrà indicata semplicemente con Q_i.
Dal momento che $x(t)$ e $u(t)$ sono ad ogni istante vettori canonici, il funzionale di costo (4.3) può essere riscritto linearmente nel seguente modo:

$$J_T(x_0, u(\cdot)) = c_f^T x(T) + \sum_{t=0}^{T-1} c^T \ltimes u(t) \ltimes x(t) \,, \tag{4.4}$$

dove

$$c_f^T = \begin{bmatrix} [Q_f]_{11} & [Q_f]_{22} & \cdots & [Q_f]_{2^n 2^n} \end{bmatrix},$$

mentre

$$\begin{aligned} c^T &= \left[\begin{array}{ccc|c|ccc|c} [Q_1]_{11} & \cdots & [Q_1]_{2^n 2^n} & \cdots & [Q_{2^m}]_{11} & \cdots & [Q_{2^m}]_{2^n 2^n} \end{array}\right] = \\ &= \left[\begin{array}{c|c|c} c_1 & \cdots & c_{2^m} \end{array}\right] . \end{aligned} \tag{4.5}$$

Osservazione 4.10: L'ipotesi che le matrici di costo Q_f e Q_i, $1 \leq i \leq 2^m$ siano semidefinite positive comporta unicamente che i vettori c_f e c siano non negativi. □

Per la soluzione del problema risulta utile una modifica del funzionale di costo a livello di scrittura, Per ogni scelta della famiglia di vettori $m(t) \in \mathbb{R}^{2^n}$, $0 \leq t \leq T$ e per ogni traiettoria di $x(t)$, si ha

$$\sum_{t=0}^{T-1} [m(t+1)^T x(t+1) - m(t)^T x(t)] + m(0)^T x(0) - m(T)^T x(T) = 0 ,$$

e perciò il funzionale di costo può essere riscritto come segue:

$$J_T(x_0, u(\cdot)) = m(0)^T x(0) + [c_f - m(T)]^T x(T) + \sum_{t=0}^{T-1} [c^T \ltimes u(t) \ltimes x(t)] +$$

$$+ \sum_{t=0}^{T-1} [m(t+1)^T x(t+1) - m(t)^T x(t)] .$$

Sfruttando l'equazione di evoluzione della BCN e la possibilità di riscrivere il prodotto $m(t)^T x(t)$ come

$$\left[\begin{array}{ccc} m(t)^T & m(t)^T & \cdots & m(t)^T \end{array}\right] \ltimes u(t) \ltimes x(t) ,$$

il funzionale di costo può essere riscritto allora come

$$J_T(x_0, u(\cdot)) = m(0)^T x(0) + [c_f - m(T)]^T x(T) +$$

$$+ \sum_{t=0}^{T-1} (c^T + m(t+1)^T L - \left[\begin{array}{ccc} m(t)^T & m(t)^T & \cdots & m(t)^T \end{array}\right]) \ltimes u(t) \ltimes x(t)].$$

A questo punto si sfrutta l'arbitrarietà di $m(t)$ per scegliere la famiglia di vettori $m(t)$, $0 \leq t \leq T$, in accordo con l'Algoritmo 4.2. Con questa scelta nel funzionale di costo il termine $[c_f - m(T)]^T x(t)$ è pari a 0, mentre il termine

$$w(t) := \left[\begin{array}{cccc} w_1(t)^T & w_2(t)^T & \ldots & w_{2^m}(t)^T \end{array}\right] =$$

$$= \left[\begin{array}{cccc} c_1^T & c_2^T & \ldots & c_{2^m}^T \end{array}\right] + m(t+1)^T \left[\begin{array}{cccc} L_1 & L_2 & \ldots & L_{2^m} \end{array}\right] -$$

Algoritmo 4.2 Algoritmo per la scelta della famiglia di vettori $m(t)$.

Input: c_f; c; L;
Output: $m(t)$, famiglia di vettori;

$m(T) = c_f$;
for $t = T-1, \ldots, 0$ **do**
  for $i = 1, \ldots, 2^n$ **do**
    $[m(t)]_i = \min\limits_{1 \le j \le 2^m} \left([c_j]_i + [m(t+1)^T L_j]_i \right)$;
  end for
end for

$$- \begin{bmatrix} m(t)^T & m(t)^T & \ldots & m(t)^T \end{bmatrix}$$

è non negativo, e per ogni i, $1 \le i \le 2^n$, esiste almeno un j, $1 \le j \le 2^m$, per il quale $[w_j(t)]_i = 0$.
Di conseguenza, il funzionale di costo

$$J_T(x_0, u(\cdot)) = m(0)^T x(0) + \\ + \sum_{t=0}^{T-1} \begin{bmatrix} w_1(t)^T & w_2(t)^T & \ldots & w_{2^m}(t)^T \end{bmatrix} \ltimes u(t) \ltimes x(t)$$

è minimizzato dalla sequenza di ingresso generata con la seguente regola:

$$x(t) = \delta_{2^n}^i \;\Rightarrow\; u(t) = \delta_{2^m}^{j^*(i,t)} ,$$

con

$$j^*(i,t) = \operatorname*{argmin}_{1 \le j \le 2^m} \left([c_j]_i + [m(t+1)^T L_j]_i \right) = \operatorname*{argmin}_{1 \le j \le 2^m} \left([w_j(t)]_i \right) .$$

Con la precedente scelta del valore dell'ingresso all'istante t, si ha quindi che

$$\begin{bmatrix} w_1(t)^T & w_2(t)^T & \ldots & w_{2^m}(t)^T \end{bmatrix} \ltimes u(t) \ltimes x(t) = 0, \; \forall \, 0 \le t \le T ,$$

che risulta essere il valore minimo che il termine può assumere, data la non negatività del vettore $w(t)$.
In definitiva, il valore minimo di costo che si può ottenere con una arbitraria sequenza di ingresso è

$$J_T^*(x_0) = \min_{u(\cdot)} J_T(x_0, u(\cdot)) = m(0)^T x(0) ,$$

e il controllo ottimo può essere implementato grazie ad una retroazione dallo stato tempo variante $u(t) = K(t)x(t)$, con

$$K(t) = \left[\begin{array}{cccc} \delta_{2^m}^{j^*(1,t)} & \delta_{2^m}^{j^*(2,t)} & \cdots & \delta_{2^m}^{j^*(N,t)} \end{array} \right] .$$

Esempio 4.11: Si consideri la BCN identificata dalla seguente matrice di transizione

$$L = \delta_8 \left[\begin{array}{cccccccc|cccccccc} 2 & 3 & 1 & 6 & 4 & 8 & 5 & 8 & 3 & 7 & 4 & 4 & 6 & 3 & 7 & 7 \end{array} \right] ,$$

in cui i vettori di costo c e c_f sono

$$c = \left[\begin{array}{cccccccc|cccccccc} 1 & 2 & 7 & 2 & 0 & 3 & 2 & 0 & 0 & 0 & 6 & 1 & 0 & 3 & 10 & 2 \end{array} \right]$$

$$c_f = \left[\begin{array}{cccccccc} 10 & 2 & 15 & 0 & 0 & 1 & 3 & 0 \end{array} \right] .$$

A questa rete è associato il grafo di stato presente in Figura 4.3, in cui gli archi sono etichettati con i costi relativi all'utilizzo di quell'arco. Se lo

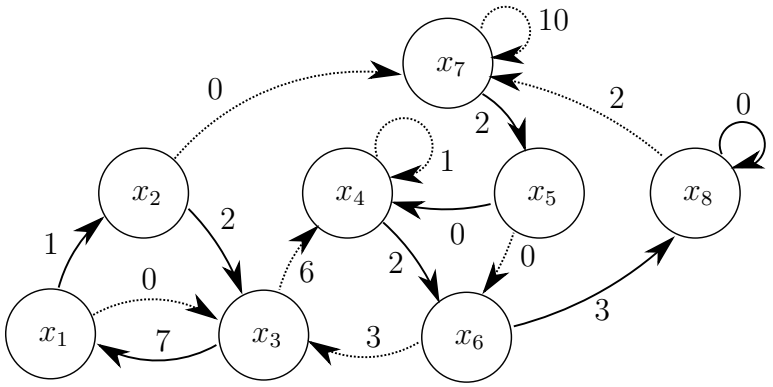


Figura 4.3: Grafo di rete relativo alla BCN dell'esempio 4.11.

scopo del controllo ottimo è quello di minimizzare il funzionale di costo (4.4) in $T = 5$ passi, si ha che l'Algoritmo 4.2 restituisce la seguente sequenza per $m(t)$

$$m(5) = c_f; \quad m(4) = \left[\begin{array}{cccccccc} 3 & 3 & 6 & 1 & 0 & 3 & 2 & 0 \end{array} \right];$$

$$m(3) = \left[\begin{array}{cccccccc} 4 & 2 & 7 & 2 & 1 & 3 & 2 & 0 \end{array} \right]; \quad m(2) = \left[\begin{array}{cccccccc} 3 & 2 & 8 & 3 & 2 & 3 & 3 & 0 \end{array} \right];$$

$$m(1) = \left[\begin{array}{cccccccc} 3 & 3 & 9 & 4 & 3 & 3 & 4 & 0 \end{array} \right]; \quad m(0) = \left[\begin{array}{cccccccc} 4 & 4 & 10 & 5 & 3 & 3 & 5 & 0 \end{array} \right].$$

La sequenza di matrici di retroazione per il controllo ottimo è

$$K(0) = \delta_2 \left[\begin{array}{cccccccc} 1 & 2 & 1 & 1 & 2 & 1 & 1 & 1 \end{array} \right];$$

$$K(1) = K(2) = K(3) = K(4) = K(5) = \delta_2 \left[\begin{array}{cccccccc} 1 & 2 & 2 & 2 & 1 & 1 & 1 & 1 \end{array} \right],$$

grazie alla quale, se lo stato iniziale è uno fra $\delta_8^1, \delta_8^2, \delta_8^3, \delta_8^7$, lo stato finale sarà δ_8^4, mentre per i rimanenti stati, lo stato finale sarà δ_8^8. Ovviamente il costo minimo a partire dallo stato δ_8^i, $1 \leq i \leq 8$ è pari a $[m(0)]_i$.

♣

4.3.2 Funzionali di costo alternativi

In questo paragrafo si affronta la soluzione di problemi di controllo ottimo ad orizzonte finito formulati in maniera diversa da quella della definizione 4.2, ma che possono essere ricondotti allo schema precedente attraverso piccole modifiche.

- **Funzionale di costo non quadratico:** il fatto che il funzionale di costo (4.3) sia quadratico, non è necessario per poter scrivere la funzione di costo in maniera lineare; se il funzionale di costo è una arbitraria funzione dell'ingresso e dello stato ad ogni istante t, è possibile riscrivere il funzionale di costo in maniera lineare. Per ogni scelta delle mappe $\mathcal{Q}_f : \Delta_{2^n} \to \mathbb{R}$ e $\mathcal{Q} : \delta_{2^m} \times \delta_{2^n} \to \mathbb{R}$, la funzione costo

$$J_T(x_0, u(\cdot)) = \mathcal{Q}_f(x(T)) + \sum_{t=0}^{T-1} \mathcal{Q}(u(t), x(t))$$

  può essere espressa come nella formula (4.4), grazie ai seguenti vettori:

$$c_f^T = \left[\begin{array}{cccc} \mathcal{Q}_f(\delta_{2^n}^1) & \mathcal{Q}_f(\delta_{2^n}^2) & \cdots & \mathcal{Q}_f(\delta_{2^n}^{2^n}) \end{array} \right]$$

$$c^T = \left[\begin{array}{cccc} c_1 & c_2 & \cdots & c_{2^m} \end{array} \right], \qquad [c_j]_i = \mathcal{Q}(\delta_{2^m}^j, \delta_{2^n}^i).$$

  Se nel funzionale di costo la matrice $Q_{u(t)}$ è tempo variante, è facile convincersi che si può arrivare a scrivere i termini della sommatoria come

$$\left[\begin{array}{cccc} c_1^T(t) & c_2^T(t) & \ldots & c_{2^m}^T(t) \end{array} \right].$$

  A questo punto è sufficiente applicare nuovamente la tecnica esposta.

- **Controllo ottimo di Mayer:** la soluzione trovata per il problema del controllo ottimo a orizzonte finito può essere applicata al problema trattato in [8] e [9]. In questi due lavori è esaminato il

problema di controllo ottimo di Mayer, che ha l'obiettivo di massimizzare un indice di qualità che pesa unicamente lo stato finale, cioè di massimizzare una funzione del tipo

$$J_T(x_0, u(\cdot)) = c_f^T x(T),\ c_f \in \mathbb{R}^{2^n}\ .$$

Se l'obiettivo fosse la minimizzazione di un indice di costo che pesa soltanto lo stato finale, ovviamente la soluzione trovata in precedenza potrebbe essere usata anche per questo problema, ponendo $c = 0$.
Per ottenere invece la massimizzazione di un indice di qualità, è sufficiente modificare l'algoritmo per la scelta della famiglia di vettori $m(t)$, prendendo, per $t = T$, $m(T) = c_f$, mentre per $0 \leq t \leq T-1$

$$[m(t)]_i := \max_{1 \leq j \leq 2^m} [m(t+1)^T L_j]_i,\ \forall\, 1 \leq j \leq 2^n.$$

In questo modo,

$$\begin{aligned} w(t) &= m(t+1)^T L - \left[\begin{array}{cccc} m(t)^T & m(t)^T & \cdots & m(t)^T \end{array}\right] = \\ &= \left[\begin{array}{c|c|c|c} w_1(t)^T & \cdots & w_{2^m}(t)^T \end{array}\right]\ , \end{aligned}$$

è un vettore che risulta sempre non positivo e con almeno un elemento pari a 0 per ogni vettore $w_j(t)$, $1 \leq j \leq 2^m$. Con la famiglia di $m(t)$ scelta, l'indice di qualità coincide con

$$J_T(x_0, u(\cdot)) = m(0)^T x(0) + \sum_{t=0}^{T-1} w(t) \ltimes u(t) \ltimes x(t)\ ,$$

e se al tempo t lo stato è $\delta_{2^n}^i$, scegliendo come ingresso $u(t) = \delta_{2^m}^{j^\star(i,t)}$, con

$$j^\star(i,t) = \operatorname*{argmax}_{1 \leq j \leq 2^m} [w_j(t)]_i\ ,$$

si ottiene la massimizzazione dell'indice di qualità.
La soluzione al problema proposta in [9] (lavoro che racchiude [8] come caso particolare) risulta più laboriosa di quella proposta in [7], perché non fornisce un metodo diretto per la costruzione della sequenza di ingresso. Quest'ultima viene costruita a ritroso a partire dall'ultimo ingresso da dare alla rete e, attraverso una serie di casistiche sugli stati, si perviene all'ingresso ottimo per il controllo.

- **Vincoli su stati e/o ingressi:** il controllo ottimo a orizzonte finito può anche essere sfruttato per imporre vincoli sul valore dello

stato finale, oppure per fare in modo che alcuni stati e/o ingressi non vengano presi in considerazione nell'evoluzione del sistema, perchè indesiderabili. Per esempio, per fare in modo che lo stato finale sia $\delta_{2^n}^i$ è sufficiente porre a $+\infty$ tutte le componenti di c_f, ad eccezione dell'elemento i-esimo; se si vuole evitare di passare durante l'evoluzione attraverso un dato stato $\delta_{2^n}^i$, è sufficiente porre $[c_j]_i = +\infty$, $\forall\, 1 \leq j \leq 2^m$, mentre se ad essere indesiderato è il ricorso ad un dato ingresso $\delta_{2^m}^k$ quando lo stato è pari a $\delta_{2^n}^i$, è sufficiente porre $[c_k]_i = +\infty$.

- **Costi sul cambio dell'ingresso:** è possibile riportare ad una forma risolvibile con il metodo esposto in precedenza un problema di controllo ottimo in cui la funzione costo pesa anche i cambiamenti nella sequenza di ingresso. La funzione di peso è la seguente:

$$\begin{aligned} J_T(x_0, u(-1), u(\cdot)) = c_f^T x(T) + \sum_{t=0}^{T-1} c^T \ltimes u(t) \ltimes x(t) + \\ + \sum_{t=0}^{T-1} p^T \ltimes u(t) \ltimes u(t-1)\,, \end{aligned} \tag{4.6}$$

nella quale il vettore $p \in \mathbb{R}^{2^{2m}}$ attribuisce pesi differenti a cambiamenti diversi sull'ingresso. Ovviamente per effettuare questo tipo di controllo è necessario disporre, ad ogni istante t, dello stato in t e dell'ingresso all'istante precedente. Per riportare il problema in una forma risolvibile con il metodo proposto, si può creare una variabile di stato aumentata

$$\xi(t) := x(t) \ltimes u(t-1)\,,$$

di cui si conosce il valore iniziale $\xi(0) = x(0) \ltimes u(-1)$. Se con L_{ij} si indica la colonna j della sottomatrice L_i, cioè

$$L_{ij} := \mathrm{Col}_j(L_i) = L \ltimes \delta_{2^m}^i \ltimes \delta_{2^n}^j,\ 1 \leq i \leq 2^m,\ 1 \leq j \leq 2^n\,,$$

e si introduce la matrice

$$\begin{aligned} \tilde{L} = \big[\ L_{11} \ltimes \delta_{2^m}^1 \mathbf{1}_{2^m}^T \quad \cdots \quad L_{12^n} \ltimes \delta_{2^m}^1 \mathbf{1}_{2^m}^T \ \big|\ \cdots \\ \big|\ L_{2^m 1} \ltimes \delta_{2^m}^{2^m} \mathbf{1}_{2^m}^T \quad \cdots \quad L_{2^m 2^n} \ltimes \delta_{2^m}^{2^m} \mathbf{1}_{2^m}^T \ \big]\,, \end{aligned}$$

si ricava l'equazione di aggiornamento dello stato aumentato

$$\xi(t+1) = \tilde{L} \ltimes u(t) \ltimes \xi(t)\,.$$

Ponendo allora

$$p^T = \left[\begin{array}{cccc} p_1^T & p_2^T & \cdots & p_{2^m}^T \end{array} \right] ,$$

con $p_i^T = p^T \ltimes \delta_{2^m}^i$, $1 \leq i \leq 2^m$, e

$$\tilde{c}_f^T = c_f^T \otimes \mathbf{1}_{2^m}^T ,$$

$$\tilde{c}^T = c^T \otimes \mathbf{1}_{2^m}^T + \left[\begin{array}{ccc|c|ccc} p_1^T & \cdots & p_1^T & \cdots & p_{2^m}^T & \cdots & p_{2^m}^T \end{array} \right] ,$$

si può dimostrare che la funzione di costo (4.6) può essere riscritta nella forma

$$J_T(\xi(0), u(\cdot)) = \tilde{c}_f^T \xi(T) + \sum_{t=0}^{T-1} \tilde{c}^T \ltimes u(t) \ltimes \xi(t) .$$

A questo punto è possibile applicare il metodo del Paragrafo 4.3.1, utilizzando i nuovi vettori di costo e $\tilde{L}$ come matrice di evoluzione dello stato.

4.3.3 Orizzonte infinito

Definizione 4.3: Data una BCN (3.3) con stato iniziale $x(0) = x_0 \in \Delta_{2^n}$, il problema del *controllo ottimo ad orizzonte infinito* consiste nel determinare una sequenza infinita di ingresso che minimizzi il funzionale di costo

$$J(x_0, u(\cdot)) = \sum_{t=0}^{+\infty} c^T \ltimes u(t) \ltimes x(t) , \tag{4.7}$$

dove $c^T \geq 0$ è un vettore appartenente a $\mathbb{R}^{2^n 2^m}$.

Come è del resto intuitivamente ovvio, si dimostra che l'unico caso in cui il costo finale risulta finito, cioè

$$J^*(x_0) := \min_{u(\cdot)} J(x_0, u(\cdot)) < +\infty ,$$

è quando esiste una traiettoria periodica di costo 0 che sia raggiungibile da x_0. Deve quindi esistere almeno un ciclo di lunghezza $T > 0$

$$\mathscr{C} = ((x(t), u(t)), (x(t+1), u(t+1)), \ldots, (x(t+T-1), u(t+T-1)))$$

tale che

$$c^T \ltimes u(k) \ltimes x(k) = 0, \ \forall\, t \leq k \leq t+T-1 .$$

Proposizione 4.2 *$J^*(x_0)$ risulta finito per ogni scelta dello stato iniziale $x_0 \in \Delta_{2^n}$ se e solo se per ogni x_0 esiste una sequenza di ingresso che a partire da un certo istante finito $\tau \geq 0$, rende la traiettoria stato-ingresso periodica e di costo zero.*

Dimostrazione: La sufficienza della condizione è evidente: se da un certo istante τ in poi il sistema resta confinato in un ciclo di costo 0, il costo totale non aumenterà più, per cui assumer certamente un valore finito. Per la necessità della condizione si può procedere come segue: tenuto conto che le coppie $(x(t), u(t))$ sono in numero finito, se $J^*(x_0)$ risulta finito, deve necessariamente esistere un qualche $\tau_1 > 0$ tale che

$$c^T \ltimes u(t) \ltimes x(t) = 0\,, \ \forall\, t \geq \tau_1\,.$$

Allo stesso tempo, sempre perché le coppie $(x(t), u(t)),\ t \geq 0$ sono in numero finito, devono esistere τ e T, con $\tau \geq \tau_1$, tali che

$$(x(\tau), u(\tau)) = (x(\tau + T), u(\tau + T))\,.$$

Allora la traiettoria

$$(\tilde{x}(t), \tilde{u}(t)) = \begin{cases} (x(t), u(t)) & 0 \geq t \geq \tau + T - 1 \\ (x(t-T), u(t-T) & \tau + T \geq t < \infty \end{cases}\,,$$

risulta essere periodica, di costo 0 e raggiungibile da x_0, per cui il teorema è stato dimostrato. ♢

Risulta quindi necessario individuare le eventuali traiettorie stato-ingresso periodiche di costo 0, e controllare che sia possibile raggiungerne almeno una a partire da ogni stato della rete.
Per stabilire l'esistenza di traiettorie periodiche di costo nullo si può procedere nel modo seguente: se

$$c^T = \begin{bmatrix} c_1 & c_2 & \cdots & c_{2^n} \end{bmatrix}$$

è il vettore riga dei costi definito in (4.5), si definiscono inizialmente 2^m matrici $C_i^{(0)}$, quadrate diagonali di dimensione 2^n, tali che

$$\mathrm{Col}_j(C_i^{(0)}) = \begin{cases} \delta_{2^n}^j & \text{se } [c_i]_j = 0\,, \\ 0 & \text{altrimenti}\,. \end{cases} \quad 1 \leq j \leq 2^n\,.$$

In questo modo il prodotto $L_i C_i^{(0)}$ fornisce una matrice quadrata di dimensione 2^n che presenta come colonne diverse da 0 solo le colonne di L_i che corrispondono a transizioni di costo 0. Preso allora

$$L^{(0)} := (L_1 C_1^{(0)}) \vee (L_2 C_2^{(0)}) \vee \cdots \vee (L_{2^n} C_{2^n}^{(0)})$$

si ha che $[L^{(0)}]_{ij} = 1$, $1 \leq i, j \leq 2^n$, se e solo se esiste un ingresso $\delta_{2^m}^k$, $1 \leq k \leq 2^m$, tale che

$$\delta_{2^n}^i = L \ltimes \delta_{2^m}^k \ltimes \delta_{2^n}^j \qquad \text{e} \qquad c^T \ltimes \delta_{2^m}^k \ltimes \delta_{2^n}^j = 0\,,$$

ossia un ingresso che fa passare a costo nullo dallo stato $\delta_{2^n}^j$ allo stato $\delta_{2^n}^i$.
Se si costruisce il grafo relativo alla matrice $L^{(0)}$, quindi un grafo di 2^n nodi in cui esiste l'arco che parte dal nodo i e arriva al nodo j se e solo se l'elemento in posizione (j, i) della matrice $L^{(0)}$ è pari a 1, è evidente che questo grafo comprende tutti e soli gli archi di costo 0 della rete, per cui esiste almeno un ciclo di costo 0 se e solo questo grafo presenta almeno un ciclo, e questo accade se e solo se $L^{(0)}$ non è nilpotente.
Se $L^{(0)}$ non è nilpotente bisogna individuare gli stati che appartengono ai cicli di costo nullo, la cui esistenza è assicurata. Introducendo l'insieme

$$H = \{\delta_{2^n}^h \in \Delta_{2^n} \mid \exists\, k \in [1, 2^n] \,:\, \mathrm{diag}[(L^{(0)})^k]\delta_{2^n}^h \neq 0\}\,,$$

questo contiene gli stati dei cicli di costo 0. Se infatti $\delta_{2^n}^h$ appartiene a H, allora esiste un k per cui si verifica $\mathrm{diag}[(L^{(0)})^k]\delta_{2^n}^h \neq 0$, il che corrisponde a dire che in k passi lo stato h-esimo torna in se stesso, per cui $\delta_{2^n}^h$ appartiene ad almeno un ciclo (e questo ciclo ha lunghezza k). A questo punto per minimizzare l'indice di costo (4.7), nel caso di stati che non appartengono ad H è sufficiente determinare la traiettoria stato-ingresso di costo minimo che permette di raggiungere uno stato di H, mentre per gli stati di H basta utilizzare la sequenza di ingressi che permette di mantenere il sistema sulla traiettoria periodica di costo 0 a cui appartiene. Ovviamente il costo minimo $J^*(x_0)$ è pari a 0 se $x_0 \in H$, mentre risulta non negativo per tutti gli altri stati.
Per calcolare il costo minimo si sfrutta nuovamente un indice di costo a tempo finito, facendo poi tendere all'infinito l'istante T; si consideri allora il seguente funzionale di costo

$$J_T^*(x_0) := \min_{u(\cdot)} \sum_{t=0}^{T-1} c^T \ltimes u(t) \ltimes x(t)$$

per una BCN (3.3) che rispetti le condizioni della Preposizione 4.2. È possibile dimostrare il seguente

Lemma 4.1 *La sequenza di vettori $\{m(t)\}_{t=T,T-1,\ldots,0}$, generata dall'Algoritmo 4.2 usando come vettore dei costi per lo stato finale $c_f = 0$, soddisfa le seguenti disuguaglianze*

$$0 = m(T) \leq m(T+1) \leq \cdots \leq m(1) \leq m(0)\,.$$

Inoltre, se T è sufficientemente grande, allora esiste $\Delta \geq 0$ per il quale

$$m^* := m(T-\Delta) = m(T-\tau),\ \Delta \leq \tau \leq T\,,$$

cioè $m(t)$ converge al vettore non negativo m^* in un umero finito di passi.

Dimostrazione: Per ogni scelta dello stato iniziale $x_0 = \delta_{2^n}^i$, $1 \leq i \leq 2^n$, e per ogni $\tau > 1$, si ha

$$\begin{aligned} m(T-\tau)^T \delta_{2^n}^i &= \min_{u(\cdot)} \sum_{t=T-\tau}^{T-1} c^T \ltimes u(t) \ltimes x(t) \\ &= \min_{u(\cdot)} \sum_{t=0}^{\tau-1} c^T \ltimes u(t) \ltimes x(t) \\ &\leq \min_{u(\cdot)} \sum_{t=0}^{\tau} c^T \ltimes u(t) \ltimes x(t) \\ &= \min_{u(\cdot)} \sum_{t=T-\tau-1}^{T-1} c^T \ltimes u(t) \ltimes x(t) \\ &= m(T-\tau-1)^T \delta_{2^n}^i \end{aligned}$$

e, poichè x_0 è generico, si ha $m(T-\tau) \leq m(T-\tau-1)$.
Se d_i è il costo totale per portare lo stato $\delta_{2^n}^i$ ad uno stato $\delta_{2^n}^h \in H$ senza passare 2 volte per uno stesso stato, allora per ogni $\tau \geq N$ si ha

$$[m(T-\tau)]_i = m(T-\tau)^T \delta_{2^n}^i = \min_{u(\cdot)} \sum_{t=0}^{\tau-1} c^T \ltimes u(t) \ltimes x(t) \leq d_j\,.$$

Di conseguenza $m(T-\tau)$ è un vettore superiormente limitato, dal momento che

$$m(T-\tau)^T \leq \begin{bmatrix} d_1 & d_2 & \dots & d_{2^n} \end{bmatrix};$$

resta quindi da dimostrare che in un numero finito di passi si raggiunge il valore limite della sequenza. Preso c_{min} pari al più piccolo elemento di c maggiore di 0, visto che $[m(T-\Delta)]_i$ è il costo minimo sull'intervallo di lunghezza Δ a partire dallo stato $\delta_{2^n}^i$, la sequenza

$$0 = [m(T)]_i \leq [m(T-1)]_i \leq \cdots \leq [m(1)]_i \leq [m(0)]_i$$

ad ogni passo o resta invariata o aumenta almeno di c_{min}. Allo stesso tempo, se la sequenza resta costante per $N+1$ istanti consecutivi, cioè

$$[m(T-\tau)]_j = [m(T-\tau-1)]_j = \cdots = [m(T-\tau-N)]_j\,,$$

allora è stato necessariamente raggiunto (al massimo all'istante $\tau+N-1$) uno stato di un ciclo di costo 0, per cui da quel momento il costo non aumenterà più. Ogni elemento della sequenza è quindi superiormente limitato, ad ogni istante o cresce o resta costante, e se resta costante per N passi consecutivi allora è necessariamente stato raggiunto il suo valore massimo: grazie a questo si può affermare che in un numero finito di passi ogni elemento della sequenza $m(t)$ raggiunge il suo valore massimo. $\diamondsuit$

Immediata conseguenza del Lemma è il seguente

Teorema 4.7 *Se, data una BCN (3.3), il relativo insieme H è diverso dall'insieme nullo e se per ogni $x_0 \in \Delta_{2^n}$ esiste almeno uno stato $\delta_{2^n}^h \in H$ raggiungibile da x_0, allora*

- *esiste $\bar{T} \geq 0$ tale che $\forall\, x_0 \in \Delta_{2^n}$*

$$J_T^*(x_0) = J_{\bar{T}}^*(x_0) = (m^*)^T x_0, \quad \forall\, T \geq \bar{T}$$

per cui

$$J^*(x_0) = \min_{u(\cdot)} \sum_{t=0}^{+\infty} c^T \ltimes u(t) \ltimes x(t) = (m^*)^T x_0\,;$$

- *m^* si ottiene come soluzione dell'Algoritmo 4.2, con $c_f = 0$, ed è un punto fisso dell'algoritmo stesso, cioè*

$$[m^*]_i = \min_{1\leq j\leq 2^m} [c_j^T + (m^*)^T L_j]_i\,, \quad 1 \leq i \leq 2^n\,, \tag{4.8}$$

che è l'equivalente per le BCN dell'equazione algebrica di Riccati per sistemi lineari;

- *definendo*

$$j^*(i) := \operatorname*{argmin}_{1\leq j\leq 2^m} [c_j^T + (m^*)^T L_j]_i\,,$$

l'ingresso per il controllo ottimo può essere implementato con una retroazione dallo stato $u(t) = Kx(t)$, con la seguente matrice di retroazione (che non risulta necessariamente unica)

$$K = \left[\begin{array}{cccc} \delta_{2^m}^{j^*(1)} & \delta_{2^m}^{j^*(2)} & \cdots & \delta_{2^m}^{j^*(2^n)} \end{array}\right].$$

L'equazione (4.8) presenta un numero infinito di soluzioni non negative, e si può dimostrare, come è fatto in [7], che è valida la seguente

Proposizione 4.3 *Nelle stesse ipotesi del Teorema 4.7, se m è una soluzione non negativa di (4.8), allora $m \geq m^*$.*

Per ricavare m^* è quindi possibile trovare la soluzione limite dell'Algoritmo 4.2, oppure trovare la più piccola soluzione non negativa di (4.8). Seppure il primo metodo assicura la convergenza dell'algoritmo al vettore m^* cercato, la convergenza può avvenire in tempi molto lunghi. Il secondo metodo può risultare invece più utile perché non è necessario calcolare tutte le soluzioni non negative di (4.8); infatti per ogni $\delta_{2^n}^h \in H$, $[m^*]_h = 0$, perchè il costo minimo per uno stato di H è nullo dal momento che appartiene lui stesso ad un ciclo di costo 0, e viceversa si dimostra che m^* è l'unica soluzione di (4.8) che prende valore 0 in corrispondenza di stati di H. È quindi possibile determinare H, impostare a 0 tutti gli elementi corrispondenti in m^* e poi trovare la soluzione di (4.8) avente quegli 0.

Proposizione 4.4 *Nelle stesse ipotesi del Teorema 4.7, se m è una soluzione di (4.8) e $[m]_h = 0 \ \forall \ h \mid \delta_{2^n}^h \in H$, allora $m = m^*$.*

Dimostrazione: Definito $I = \{i | \delta_{2^n}^i \in H\}$, poichè $[m^*]_h = 0 \ \forall \ h \in I$, si dimostra che tutti gli elementi di m di indice $k \notin I$ sono univocamente determinati dal valore assunto da m in corrispondenza agli indici $i \in I$; in questo modo tutti i vettori m che sono nulli in corrispondenza agli indici di I coincidono con m^*. Si denoti con H_k l'insieme contenente gli indici $j \in \{1, \ldots, 2^n\}$ degli stati $\delta_{2^n}^j$ per i quali il cammino di costo minimo che porta il sistema da $\delta_{2^n}^j$ ad uno stato di H ha lunghezza k; per ogni $j \in H_1$ esiste un ingresso $\delta_{2^m}^q$ per il quale $L \ltimes \delta_{2^m}^h \ltimes \delta_{2^n}^j = \delta_{2^n}^h \in H$, e

$$[m]_j = [c_q^T + m^T L_q]_j = [c_q]_j + [m]_h = [c_q]_j \ ,$$

per cui per ogni $j \in H_1$ l'elemento j-esimo di m è univocamente determinato.
A questo punto si può pensare che H_2 contenga gli indici degli stati per i quali il percorso di costo minimo per raggiungere un qualche stato $\delta_{2^n}^i$, $i \in H_1$, ha lunghezza 1, e si può applicare il ragionamento precedente ottenendo il risultato desiderato. $\diamondsuit$

Per l'implementazione in Matlab di una funzione per ottenere il controllo ottimo ad orizzonte infinito, è stato elaborato un algoritmo che non ricava m^* attraverso l'equazione (4.8), ma che sfrutta invece l'idea dell'algoritmo di Dijkstra per i grafi ([27]), il cui scopo è quello di identificare il minimo cammino a partire da uno stato per raggiungere un qualsiasi altro stato del grafo.

Una volta costruito l'insieme H, come è già stato evidenziato, risulta sufficiente identificare il percorso di costo minimo che conduce un qualsiasi stato $x \in \Delta_{2^n} \backslash H$ in uno stato di H, e per questo è stato creato l'Algoritmo 4.3 che ha un forte legame con l'algoritmo di Dijkstra sopra citato.

Algoritmo 4.3 Algoritmo per l'identificazione del percorso minimo per raggiungere cicli di costo 0.

Input: $H;\ c;\ L;$

Output: I_{co}, matrice che nella colonna i contiene l'ingresso da dare in corrispondenza allo stato $\delta_{2^n}^i$; c_{tot}, vettore contenente i costi totali da ogni stato.

$S = H;$
for $\forall\ \delta_{2^n}^i\ \in\ H$ **do**
$[c_{tot}]_i = 0;$
$\bar{u}$= ingresso di costo 0 per rimanere nel ciclo di costo nullo a cui appartiene;
$\mathrm{Col}_i(I_{co}) = \bar{u};$
end for
while $|S| \neq 2^n$ **do**
$[v\ h] = \underset{i \in S,\ j \in \Delta_{2^n} \backslash S}{\operatorname{argmin}} \{[c_{tot}]_i + [c_{\bar{u}}]_j \mid \exists\ \bar{u}\ \in\ \Delta_{2^m} : i = L \ltimes \bar{u} \ltimes j\};$
$[c_{tot}]_h = [c_{tot}]_v + [c_{\bar{u}}]_h;$
$\mathrm{Col}_h(I_{co}) = \bar{u};$
$S = S \cup \{h\};$
end while

Ovviamente l'algoritmo è in grado di trovare la soluzione solo se la BCN ha almeno un ciclo di costo 0 (e quindi $H \neq \varnothing$), e se da ogni stato si può raggiungere uno stato di H.
Ogni volta che l'Algoritmo 4.3 aggiunge lo stato $x_h = \delta_{2^n}^h$ all'insieme S, il costo contenuto in $[c_{tot}]_h$ del percorso dallo stato x_h ad un ciclo di costo 0 è quello minimo cercato, poichè vale il seguente

Teorema 4.8 *Se $[c_{tot}]_i$ contiene il costo minimo per raggiungere un ciclo di costo 0 a partire da ogni stato $x_i = \delta_{2^n}^i\ \in\ S$, e se*

$$[v\ h] = \underset{i \in S,\ j \in \Delta_{2^n} \backslash S}{\operatorname{argmin}} \left\{[c_{tot}]_i + [c_{\bar{u}}]_j \mid \exists\ \bar{u}\ \in\ \Delta_{2^m} : \delta_{2^n}^i = L \ltimes \bar{u} \ltimes \delta_d s^j\right\}, \tag{4.9}$$

allora $[c_{tot}]_h = [c_{tot}]_v + [c_{\bar{u}}]_h$ è il costo minimo per condurre lo stato $x_h = \delta_{2^n}^h$ ad un ciclo di costo 0.

Dimostrazione: $[c_{tot}]_v + [c_{\bar{u}}]_h$ è il costo di un cammino dallo stato x_h ad un ciclo di costo nullo, ed è dato dalla somma di $[c_{tot}]_v$, costo minimo per raggiungere un ciclo a partire da $x_v = \delta^v_{2^n} \in S$, e $[c_{\bar{u}}]_h$, costo del passaggio dallo stato x_h allo stato x_v. Considerando un qualsiasi altro cammino di lunghezza k, $P = \Big((x_v = \delta^v_{2^n}, \delta^{i_1}_{2^n}), (\delta^{i_1}_{2^n}, \delta^{i_2}_{2^n}), \ldots, (\delta^{i_{k-1}}_{2^n}, x_f = \delta^{i_k}_{2^n} \in H)\Big)$, che permette di passare da x_v a uno stato di un ciclo a costo nullo, si denoti con $x_j = \delta^{i_j}_{2^n}$, $1 \leq j \leq k-1$, l'ultimo stato attraversato nel cammino P che non appartiene ad S, cioè $x_j \in \Delta_{2^n} \backslash S$, $\delta^{i_\ell}_{2^n} \in S, j+1 \leq \ell \leq k$. Il cammino P può essere partizionato in due parti, P_1 che contiene il cammino tra x_v e $\delta^{i_{j-1}}_{2^n}$, e P_2, contenente il cammino da x_j a x_f. Denotando con $c(R)$ il costo di un cammino R del grafo, e con $\hat{u}$ l'ingresso che permette il passaggio tra x_j e $\delta^{i_{j+1}}_{2^n}$ si ha

$$c(P) = \underbrace{c(P_1)}_{\geq 0} + [c_{\hat{u}}]_{i_j} + \underbrace{c(P_2)}_{\geq [c_{tot}]_{i_{j+1}}} \geq [c_{tot}]_{i_{j+1}} + [c_{\hat{u}}]_{i_j} \geq [c_{tot}]_v + [c_{\bar{u}}]_h \, .$$

Tutta la dimostrazione si basa sul fatto che il vettore dei costi è non negativo. $\diamondsuit$

Ovviamente il vettore c_{tot} coincide necessariamente con il vettore m^*, e la matrice I_{co} conterrà gli ingressi da dare in corrispondenza di ogni stato, per cui, ponendo $K = I_{co}$ si ottiene la matrice di retroazione per ottenere il controllo ottimo ad orizzonte infinito per la BCN.

Esempio 4.12: Si consideri nuovamente la BCN dell'Esempio 4.11, in cui però il vettore di costi c sia il seguente:

$$c = \left[\begin{array}{cccccccc|cccccccc} 1 & 2 & 0 & 2 & 0 & 3 & 2 & 0 & 0 & 0 & 6 & 1 & 0 & 3 & 10 & 2 \end{array}\right] .$$

Se si vuole effettuare controllo ottimo con orizzonte infinito, bisogna inizialmente individuare tutti i cicli di costo 0 della BCN; osservando il grafo di stato (Figura 4.3, con l'unica modifica che l'etichetta dell'arco continuo uscente dallo stato 3 vale 0), è facile individuare che i cicli di interesse sono

$$\mathscr{C}_1 = (\delta^8_8, \delta^1_2); \qquad \mathscr{C}_2 = \big((\delta^1_8, \delta^2_2), (\delta^3_8, \delta^1_2)\big) \, ,$$

stessa conclusione che si ottiene calcolando $(L^{(0)})^8$ che presenta solo la prima, la terza e la ottava riga diverse da 0 e quindi $H = \{\delta^1_8, \delta^3_8, \delta^8_8\}$. Applicando l'Algoritmo 4.3, si ottiene il seguente vettore dei costi totali

$$c_{tot} = \left[\begin{array}{cccccccc} 0 & 2 & 0 & 5 & 3 & 3 & 5 & 0 \end{array}\right] ,$$

e il vettore I_{co} invece è

$$I_{co} = \delta_2 \left[\begin{array}{cccccccc} 2 & 1 & 1 & 1 & 2 & 1 & 1 & 1 \end{array} \right] .$$

Volendo verificare per conferma che $c_{tot} = m^*$, è stato anche modificato l'Algoritmo 4.2, in modo che continui a calcolare $m(t)$ finché questo non verifica (4.8), ed effettivamente si trova lo stesso risultato ottenuto con l'Algoritmo 4.3.
Con gli ingressi scelti si ottiene che gli stati δ_8^1, δ_8^2 e δ_8^3 vengono "assorbiti" dal ciclo $\mathscr{C}_2$, mentre i rimanenti vengono "assorbiti" da $\mathscr{C}_1$. In realtà la matrice I_{co} che assicura il costo minimo non è unica, perché potrebbero esserci più stati e ingressi che soddisfano contemporaneamente l'equazione (4.9); nel caso in esame infatti, anche la matrice di retroazione

$$K = \delta_2 \left[\begin{array}{cccccccc} 2 & 1 & 1 & 1 & 2 & 2 & 1 & 1 \end{array} \right]$$

assicura che il costo finale per ogni stato $x_i = \delta_{2^n}^i$ sia pari a $[c_{tot}]_i$. Con l'utilizzo di quest'ultima matrice di retroazione, solo lo stato δ_8^8 resta nel ciclo $\mathscr{C}_1$, mentre tutti i rimanenti vengono "assorbiti" da $\mathscr{C}_2$. ♣

L'ultimo aspetto sul controllo ottimo ad orizzonte infinito che viene trattato riguarda la minimizzazione di un funzionale di costo medio, cioè

$$J_a(x_0, u(\cdot)) = \lim_{T \to +\infty} \frac{1}{T} \sum_{t=0}^{T-1} c^T \ltimes u(t) \ltimes x(t) , \tag{4.10}$$

con $c \in \mathbb{R}^{2^n 2^m}$, vettore non negativo; in [1] e [10] viene esaminata una funzione di costo simile, con l'obiettivo però di massimizzarla.

Definizione 4.4: Data la BCN (3.3) con stato iniziale $x(0) = x_0 \in \Delta_{2^n}$, il problema della *minimizzazione del costo medio* consiste nel trovare la sequenza di ingresso $u(\cdot)$ che minimizza il funzionale di costo (4.10).

Considerando inizialmente un simile problema ad orizzonte finito, si ha che la stessa sequenza di ingresso $u(\cdot)$ che risolve

$$J_T^*(x_0) = \min_{u(\cdot)} \sum_{t=0}^{T-1} c^T \ltimes u(t) \ltimes x(t) ,$$

risolve anche

$$\min_{u(\cdot)} \frac{1}{T} \sum_{t=0}^{T-1} c^T \ltimes u(t) \ltimes x(t) ,$$

se lo stato iniziale è x_0 per entrambi. Denotato con $\tilde{m}_T(0)$, il valore assunto da $m(t)$ in corrispondenza dell'istante 0, con $m(t)$ calcolato grazie all'Algoritmo 4.2 con condizione iniziale $m(T) = 0$, si ha che $J_T^*(x_0) = \tilde{m}_T(0)^T x_0$, per cui

$$\min_{u(\cdot)} \frac{1}{T} \sum_{t=0}^{T-1} c^T \ltimes u(t) \ltimes x(t) = \frac{1}{T} \tilde{m}_T(0)^T x_0 \, .$$

Grazie a quest'ultima considerazione si ottiene

$$J_a^*(x_0) = \min_{u(\cdot)} J_a(x_0, u(\cdot)) = \lim_{T \to +\infty} \frac{1}{T} \tilde{m}_T(0)^T x_0 = m_a^T x_0 \, ,$$

con

$$m_a = \lim_{T \to +\infty} \frac{1}{T} \tilde{m}_T(0) \, .$$

In maniera simile a quanto accadeva per il controllo ottimo a orizzonte infinito, il minimo costo medio a partire da un dato stato iniziale x_0, si ottiene portando il sistema ad una traiettoria periodica di minimo costo medio, che ovviamente sia raggiungibile da x_0. Considerato un ciclo

$$\mathscr{C} = ((\delta_{2^n}^{i_1}, \delta_{2^m}^{j_1}), (\delta_{2^n}^{i_2}, \delta_{2^m}^{j_2}), \ldots, (\delta_{2^n}^{i_d}, \delta_{2^m}^{j_d})) \, ,$$

dove ovviamente $L \ltimes \delta_{2^m}^{j_d} \ltimes \delta_{2^n}^{i_d} = \delta_{2^n}^{i_1}$, il costo medio del ciclo è

$$C_a(\mathscr{C}) = \frac{1}{d} \sum_{h=1}^{d} c^T \ltimes \delta_{2^n}^{j_h} \ltimes \delta_{2^n}^{i_h} \, ,$$

e si ha

$$J_T^*(x_0) = \min\{C_a(\mathscr{C}_i) \mid \mathscr{C}_i \text{ raggiungibile da } x_0\} \, .$$

Per determinare quindi la sequenza per il controllo ottimo si può sfruttare il procedimento introdotto per il controllo ottimo a orizzonte finito, procedendo in questo modo:

- a partire da un ogni stato iniziale $x_0 \in \Delta_{2^n}$, vanno inizialmente individuati i cicli di costo medio minore che sono raggiungibili da x_0 e, se sono più di uno, viene scelto quello che può essere raggiunto con il minimo costo; il ciclo selezionato viene identificato con $\mathscr{C}_i(x_0)$;
- per ogni ciclo

$$\mathscr{C}_i(x_0) = ((\delta_{2^n}^{i_1}, \delta_{2^m}^{j_1}), (\delta_{2^n}^{i_2}, \delta_{2^m}^{j_2}), \ldots, (\delta_{2^n}^{i_d}, \delta_{2^m}^{j_d}))$$

si pone a 0 ogni elemento $[c_{j_h}]_{i_h}$, $1 \leq h \leq d$ di c. In questo modo, con questo nuovo vettore di costo c i cicli di costo medio minimo sono diventati cicli di costo nullo, per cui con l'algoritmo trovato per il controllo ottimo, a partire dallo stato x_0 la sequenza di ingresso calcolata porterà il sistema in uno di questi cicli;

- si identificano quindi gli stati appartenenti ad H, si costruisce un vettore m^*, che abbia obbligatoriamente $[m^*]_h = 0$, $\forall\, h | \delta_{2^n}^h \in H$ e si determinano i restanti elementi di m^* attraverso l'equazione (4.8); per determinare la matrice di retroazione K si procede come nel Teorema 4.7

In questo modo il problema di minimizzazione di una funzione di costo media è stato risolto sfruttando il procedimento elaborato per il problema del controllo ottimo.
Con questo ultimo argomento si chiude il capitolo sulle strategie di controllo di una rete Booleana.

5 Applicazione

In questo capitolo viene trattata una possibile applicazione delle strategie di controllo proposte. Come già accennato nell'introduzione, esistono numerosi sistemi biologici che possono essere modellati attraverso delle reti Booleane, ottenendo validi risultati a livello predittivo; se tradizionalmente lo strumento più usato per modellare tali sistemi sono state le equazioni differenziali, recentemente anche l'uso di reti Booleane sta prendendo piede. I sistemi biologici che finora sono stati modellati con reti Booleane sono relativi a cicli cellulari di lieviti [22], [23], a cicli cellulari dei mammiferi, [20], al ciclo di funzionamento di geni del DNA [12], nonché ad altri sistemi regolati da geni e proteine. Si è più volte osservato su base sperimentale, che se un gene risulta attivato a causa della presenza (o dell'assenza) di una determinata proteina, normalmente esso conserva il suo livello di attività per valori di concentrazione della proteina che variano anche di molti ordini di grandezza. Perciò è giustificabile il fatto di modellare i geni come variabili Booleane, con valore 0 se i geni risultano inattivi e con valore 1 se invece sono attivi, e le proteine come variabili Booleane con valore 1 se presenti in alta concentrazione e con valore 0 se presenti in bassa concentrazione. Inoltre si osserva che il cambio di concentrazione di una proteina avviene di norma molto rapidamente, per cui la semplificazione dell'uso del tempo discretizzato risulta giustificabile [21].
A differenza delle equazioni differenziali, i modelli basati su reti Booleane richiedono molti meno parametri ed è più facile verificare la loro correttezza di previsione: nel caso delle equazioni differenziali, ha molta importanza il valore della concentrazione delle sostanze, e, di conseguenza, per verificare se le simulazioni sono concordi agli esperimenti, è necessaria l'applicazione di tecniche molto più raffinate. D'altra parte, la modellizzazione tramite reti logiche non è in grado di stabilire quando temporalmente avvengono i diversi passaggi (per esempio quelli della suddivisione cellulare), ma ne fornisce solo la sequenza logica, mentre le equazioni differenziali sono in grado di fornire anche una stima temporale degli eventi. L'uso di un modello o di un altro è quindi dettato dalle esigenze e dagli obiettivi che si vogliono raggiungere, e ciascun metodo presenta dei pregi e dei difetti.
L'esempio che viene trattato qui di seguito riguarda il funzionamento dell'operone *lac*, ed è stato scelto perché risulta abbastanza facile identificare alcune variabili che possono essere considerate di ingresso, e si possono fare delle ipotesi sulle possibili variabili di uscita. In altri esempi (si vedano ad esempio [22] o [23]) non risulta facile individuare una

variabile che non sia influenzata dalle altre e che possa essere facilmente controllata dall'uomo (requisito necessario se si pensa al controllo come intervento dell'uomo su un determinato sistema).

5.1 L'operone *lac*

L'esempio trattato in questo capitolo è stato tratto da [12], e riguarda il funzionamento dell'operone *lac* presente nel batterio *Escherichia coli*. L'operone *lac* è un insieme di 3 geni che sono responsabili del metabolismo del lattosio dal parte del batterio; il suo studio è risultato molto importante per quanto riguarda la ricerca sul metabolismo degli zuccheri: in presenza di glucosio (al di fuori della cellula) questo operone non viene attivato e il batterio non metabolizza lattosio, mentre se il glucosio extracellulare è assente, l'operone si attiva e il batterio utilizza il lattosio come fonte di energia.
L'operone *lac* è inducibile (cioè si può stilarne la trascrizione), sia con un controllo negativo (cioè è presente un repressore che blocca la trascrizione genica, repressore che viene inattivato in presenza del suo induttore), sia con un controllo positivo (cioè è presente un attivatore che favorisce la la trascrizione dei geni, che però è funzionante solo in presenza del suo induttore), come sarà successivamente meglio spiegato. L'operone in esame risulta quindi uno dei primi esempi scoperti di sistemi di geni che possono essere controllati sia positivamente che negativamente.
L'operone *lac* è stato individuato e studiato per la prima volta nel 1961 [13], e un primo modello del comportamento e delle interazioni di questo sistema di geni risale al 1963 [14]. Nel tempo sono stati elaborati numerosi modelli e quasi tutti sfruttano sistemi di equazioni differenziali (per esempio [17]), anche se ne esistono alcuni a tempo discreto (vedere [12] per una lista esaustiva).
Nel seguito si riporta una sezione dedicata allo studio del funzionamento dell'operone *lac* a livello biologico e successivamente si presenta la rete Booleana elaborata in [12] per modellare questo sistema e su questo si applicano le diverse strategie di controllo dall'uscita che sono state studiate ed elaborate nel corso della tesi.

5.1.1 Modello biologico

In biologia si definisce *operone* un gruppo di geni adiacenti, che vengono trascritti insieme in una singola molecola di mRNA e che codificano per proteine aventi scopi tra loro legati (nel caso in esame i tre geni codificano per proteine che sono tutte coinvolte nella digestione del lattosio). L'ope-

rone *lac* è formato da 3 geni contigui, *Lac Z, Lac Y* e *LacA*. Il primo gene codifica per l'enzima *β-galattosidasi*, in grado di scindere il disaccaride lattosio in glucosio e galattosio; il secondo gene codifica per la *permeasi*, enzima che si lega alla membrana cellulare e permette l'ingresso nel citoplasma cellulare del lattosio; l'ultimo gene codifica per la *tiogalattoside transacetilase*, enzima che non risulta avere molta importanza nella regolazione del funzionamento del ciclo metabolico del lattosio, e che quindi viene in questo contesto tralasciato.
Nel DNA l'operone *lac* è preceduto da un *promotore*, cioè il sito a cui si lega l'enzima RNA polimerasi e da cui poi questa comincia la trascrizione dell'mRNA, e da un *operatore*, cioè un sito a cui può legarsi un repressore, una molecola in grado di bloccare fisicamente il passaggio all'RNA polimerasi e quindi di bloccare la trascrizione dell'operone. All'interno del citoplasma, è inoltre presente un'altra proteina, CAP, detta attivatore, che, quando è attiva, è in grado di favorire il legame dell'RNA polimerasi con il promotore e quindi la trascrizione dell'mRNA risulta molto più efficiente.
In assenza di glucosio extracellulare, che è la fonte di energia preferita dal batterio, e in presenza di lattosio, si attiva la trascrizione dell'operone *lac*: molecole di *allolattosio*, un derivato del lattosio (che si crea comunque in presenza della β-galattosidasi), sono in grado di legarsi al repressore dell'operone (sempre presente all'interno della cellula e codificato dal gene *LacI*); in questo modo il repressore (chiamato LacI) si stacca dall'operatore e l'RNA polimerasi è in grado di avanzare sul filamento di DNA e di trascrivere l'mRNA relativo. Inoltre, in assenza di glucosio si crea una molecola, cAMP, in grado di legarsi a CAP e quindi di attivarla; in questo modo la trascrizione dell'mRNA avviene in maniera molto più efficiente e si producono grandi quantità di β-galattosidasi che scinde il lattosio, e di permeasi, che facilita l'ingresso di ulteriore lattosio nella cellula (ingresso che può avvenire comunque ma che risulta molto lento e difficile).
La presenza di glucosio extracellulare è in grado di bloccare tutto il processo della trascrizione dell'operone *lac*, con due azioni distinte: la *repressione da catabolita* e l'*esclusione dell'induttore*. La prima azione consiste nell'ostacolare la trascrizione dell'operone *lac*: la presenza di glucosio mantiene a livelli bassissimi la concentrazione della molecola cAMP, per cui l'attivatore CAP è sempre inibito; questo comporta che se anche fosse presente dell'allolattosio nella cellula e quindi il repressore non fosse legato all'operatore, l'RNA polimerasi trascriverebbe in maniera molto inefficiente l'RNA. La seconda azione invece impedisce il trasporto del lattosio all'interno della cellula, poichè inibisce l'azione della RNA

polimerasi, e quindi non avviene fisicamente il passaggio del lattosio dall'esterno all'interno della cellula (e quindi il repressore è nuovamente in grado di attaccarsi all'operatore e quindi il processo di trascrizione si blocca).
Il fatto che l'operone *lac* sia inducibile attraverso un controllo negativo è conseguenza del fatto che in presenza di allolattosio nella cellula, il repressore LacI viene inattivato (l'allolattosio per questo si definisce un induttore), mentre è inducibile anche con un controllo positivo perchè in assenza di glucosio viene prodotta in quantità importante la molecola cAMP che funge da induttore per l'attivatore CAP, che quindi diventa funzionante. Condizione necessaria quindi per il metabolismo del lattosio è la presenza di lattosio e l'assenza di glucosio all'esterno della cellula.

5.1.2 Modello Booleano

In questo paragrafo si ricava un modello Booleano a partire dal funzionamento biologico del sistema appena illustrato. Le variabili usate per modellare il sistema sono le seguenti:

- M, rappresenta l'mRNA relativo all'operone *lac*;
- C, rappresenta l'attivatore CAP;
- R, rappresenta il repressore LacI;
- A, rappresenta l'allolattosio;
- P, rappresenta la permeasi;
- B, rappresenta la β-galattosidasi;
- L, rappresenta il lattosio intracellulare;
- L_e, rappresenta il lattosio extracellulare;
- G_e, rappresenta il glucosio extracellulare.

Le variabili M, P, C, B e G_e, possono assumere solo i valori 0 e 1 (attiva/non attiva, presente/assente), mentre per R, A, L, L_e, [12] usa in realtà 2 variabili Booleane, per tenere conto della maggiore o minore concentrazione (per poter fare un confronto con i modelli che usano equazioni differenziali); ad esempio, per rappresentare il lattosio extracellulare si usano 2 variabili, L_e e L_{em}, in questo modo:

- $L_e = 0$, $L_{em} = 0$ rappresentano una bassa concentrazione di lattosio;

- $L_e = 0$, $L_{em} = 1$ rappresentano una concentrazione di lattosio media;
- $L_e = 1$, $L_{em} = 1$ rappresenta una alta concentrazione di lattosio.

Lo stesso schema si applica a tutti gli altri elementi modellati tramite 2 variabili. Nel seguito si ricavano le equazioni logiche di aggiornamento dei nodi, basandosi sulle informazioni ricavate dal modello biologico. Se due variabili sono legate da $\wedge$, allora devono essere entrambe presenti o attive, mentre se sono legate da $\vee$ è sufficiente che solo una delle due sia presente o attiva; per indicare la negazione, e quindi che la sostanza deve essere assente o inattiva, si usa una barra sopra la variabile da negare.

- Funzione Booleana per M: affinché l'mRNA venga creato, l'attivatore CAP deve essere presente, mentre il repressore deve essere inattivo, quindi:

$$M(t+1) = C(t) \wedge \bar{R}(t) \wedge \bar{R}_m(t)\,;$$

- funzioni Booleane per P e B: per la creazione di permeasi e β-galattosidasi è necessaria la presenza dell'mRNA dell'operone *lac*, quindi:

$$P(t+1) = M(t)\,, \qquad B = M(t)\,;$$

- funzione Booleana per C: affinchè l'attivatore CAP sia presente, non deve esserci del glucosio all'esterno della cellula (è stata tralasciata nella modellazione la presenza della molecola cAMP, perché in definitiva corrisponde all'assenza del glucosio):

$$C(t+1) = \bar{G}_e(t)\,;$$

- funzione Booleana per R: nel citoplasma è presente una forte concentrazione di repressori "attivi" (cioè a cui non sono legate molecole di allolattosio) se nel citoplasma stesso non è presente allolattosio in grandi concentrazioni:

$$R(t+1) = \bar{A}(t) \wedge \bar{A}_m(t)\,;$$

- funzione Booleana per R_m: la concentrazione del repressore "attivo" è presente (almeno) a livello medio se l'allolattosio non è presente oppure se all'istante precedente la concentrazione del repressore stesso era molto elevata:

$$R_m(t+1) = (\bar{A}(t) \wedge \bar{A}_m(t)) \vee R(t)\,;$$

- funzione Booleana per A: affinchè nel citoplasma sia presente allolattosio in grandi quantità è necessaria la presenza di β-galattosidasi e di lattosio (a livello intracellulare) in elevate concentrazioni:

$$A(t+1) = B(t) \wedge L(t) \, ;$$

- funzione Booleana per A_m: per avere allolattosio in piccole quantità è sufficiente la presenza di lattosio all'interno della cellula almeno in media concentrazione (è comunque sempre presente una piccola quantità di β-galattosidasi nel batterio, anche se in quantità molto ridotta, e questa presenza permette la trasformazione del lattosio in allolattosio):

$$A_m(t+1) = L(t) \vee L_m(t) \, ;$$

- funzione Booleana per L: per avere un'alta concentrazione di lattosio intracellulare è necessaria la presenza di permeasi e di lattosio extracellulare in alta concentrazione, e contemporaneamente deve essere assente il glucosio:

$$L(t+1) = P(t) \wedge L_e(t) \wedge \bar{G}_e(t) \, ;$$

- funzione Booleana per L_m: per avere una concentrazione media di lattosio, è necessario che sia presente la permeasi e che il lattosio extracellulare sia presente in concentrazione media, oppure è sufficiente che sia presente molto lattosio all'esterno della cellula (in grado di passare all'interno della cellula per diffusione o a causa di una quantità molto ridotta di permeasi che è sempre presente come nel caso della β-galattosidasi); in ogni caso deve essere assente il glucosio all'esterno della cellula:

$$L_m(t+1) = [(L_{em}(t) \wedge P(t)) \vee L_e(t)] \wedge \bar{G}_e(t) \, .$$

Il glucosio e il lattosio extracellulari possono essere pensati come gli ingressi del sistema, perché non sono influenzati da nessuna delle altre variabili ed è anche ragionevole pensare di poterli controllare dall'esterno.
In Figura 5.1 si riporta il grafo di rete della BCN ricavata, in cui risulta evidente il fatto che G_e e L_e possono essere considerati degli ingressi del sistema.

Come si nota dal grafo gli archi sono di due tipi, ad indicare l'effetto che ha una variabile se ne influenza un'altra: se la sua presenza favorisce l'altra variabile, l'arco è una freccia, se invece la ostacola, l'arco termina

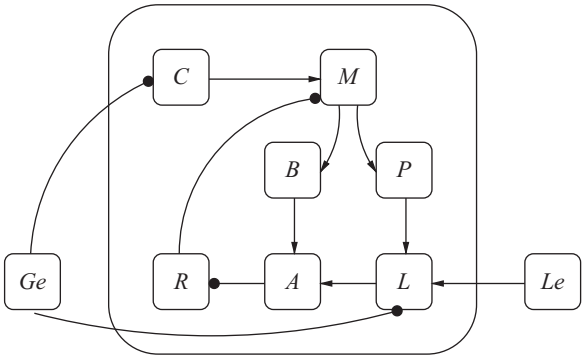


Figura 5.1: Grafo di rete relativo alla rete Booleana che modella il funzionamento dell'operone *lac*.

con un pallino (le variabili modellate con 2 elementi sono rappresentate con un solo nodo).
Nella forma algebrica, se denotiamo con un apice v la forma vettoriale delle variabili Booleane, la variabile che verrà usata nei calcoli sarà:

$$x(t) = M^v(t) \ltimes B^v(t) \ltimes R^v(t) \ltimes A^v(t) \ltimes L^v(t) \ltimes P^v(t) \ltimes C^v(t) \ltimes R^v_m(t) \ltimes \\ \ltimes A^v_m(t) \ltimes L^v_m(t) \,,$$

mentre la variabile di ingresso in forma vettoriale sarà

$$u(t) = G^v_e(t) \ltimes L^v_{em}(t) \ltimes L^v_e(t) \,.$$

Il sistema che ci si trova di fronte non presenta realmente 1024 (2^{10}) stati distinti, guidati da 8 (2^3) ingressi. Infatti le variabili rappresentate da 2 elementi non possono assumere tutte le configurazioni possibili (ad esempio per L e L_m, la configurazione $L = 1$, $L_m = 0$ è priva di senso, e questo vale anche per R e R_m, A e A_m, L_e e L_{em}): le variabili di questo tipo hanno 3 possibili configurazioni. Si avranno quindi 432 stati distinti ($3^3 \cdot 2^4$) e 6 ($3 \cdot 2$) ingressi distinti (ed è necessaria una maggiore attenzione quando si fa il passaggio dalla forma vettoriale a quella Booleana e viceversa, come sarà sottolineato successivamente).
Dopo aver ottenuto il sistema in forma algebrica, sono state analizzate le 6 reti Booleane che si ottengono fissando i valori dell'ingresso in tutti e sei i possibili modi (cioè sono stati analizzati i 6 sottosistemi della BCN), alla ricerca dei punti fissi e dei cicli, per poter fare un confronto con quelli individuati in [12]. In Tabella 5.1 si riportano i risultati ottenuti: la prima colonna contiene il valore dell'ingresso in forma vettoriale, la seconda l'ingresso esprimendo il valore delle singole variabili Booleane,

la terza e la quarta colonna contengono i punti fissi (che sono gli unici cicli dei sottosistemi) sotto forma vettoriale e come variabili logiche[3] , l'ultima colonna contiene la cardinalità del bacino di attrazione relativo ad ogni punto fisso.

I risultati ottenuti coincidono con quelli di [12], ed è facile verificare che

Ingressi (f v)	Ingressi (f l)	Punti fissi (f v)	Punti fissi (f l)	Bac. attr.
1	111	360	0010000100	432
2	101	360	0010000100	432
3	100	360	0010000100	432
4	011	38	1101111011	432
5	001	81	1100011011	8
5	001	356	0010001100	424
6	000	356	0010001100	432

Tabella 5.1: Punti fissi della rete Booleana che modella il funzionamento dell'operone *lac*.

c'è corrispondenza anche a livello biologico. Nei primi tre casi è infatti presente glucosio all'esterno della cellula, per cui il batterio non dovrebbe codificare l'operone *lac*; effettivamente l'unico punto fisso delle BN relative a questi ingressi presenta tutte le variabili nulle tranne le variabili R e R_m, quindi nella cellula è presente il repressore (che è attivo), e tutte le altre sostanze sono assenti.

Nel quarto caso è presente all'esterno della cellula una gran quantità di lattosio, mentre il glucosio è assente: il modello prevede che che siano presenti tutte le sostanze necessarie per la digestione del lattosio e che il repressore sia inattivo; questo è ovviamente quello che succede anche a livello biologico.

Tralasciando momentaneamente la BN che si ottiene con il quinto ingresso, si analizza la rete che si ottiene nel sesto caso, quando nessun tipo di zucchero (di quelli qui considerati) è presente all'esterno della cellula. Il ciclo limite previsto dal modello presenta tutte le variabili a 0 tranne le variabili R, R_m e C, quindi l'operone non è stato trascritto ma a differenza del primo caso analizzato, è presente l'attivatore CAP (attivato), fatto compatibile con l'assenza di glucosio. Il fatto che in assenza

[3]Il numero che appare nella colonna della forma vettoriale, per esempio 360, non indica il 360° stato sui 1024, ma indica il 360° stato sui 432 della rete ridotta. Per ottenere i valori delle singole variabili Booleane M,R, etc., bisogna risalire al valore dello stato rispetto a 1024 e poi applicare l'algoritmo esposto nel Capitolo 1 legato all'equazione (1.1). La corrispondenza dei punti fissi rispetto ai 1024 stati di partenza è la seguente: $360 \to 892,\ 38 \to 133,\ 81 \to 229,\ 356 \to 884$.
Un discorso analogo vale per gli ingressi.

di glucosio e lattosio l'operone non sia attivo è in linea con la politica di risparmio energetico della cellula, che non produce delle sostanze se poi non sono utili al funzionamento della cellula stessa (sempre legato a questa politica è il fatto che in presenza di glucosio l'operone sia inattivo, perchè la digestione del lattosio comporta la sua scissione in glucosio e in galattosio, per cui sarebbe inutilmente dispendioso digerire il lattosio se è già presente del glucosio che è immediatamente utilizzabile).
La rete che viene a crearsi utilizzando il quinto ingresso presenta 2 cicli, e viene quindi detta bistabile; mantenendo un livello medio di glucosio, alcuni batteri continuano a trascrivere l'operone *lac*, mentre altri interrompono la sua trascrizione (e secondo il modello creato, l'operone a regime si disattiva per un numero molto maggiore di stati iniziali diversi). Questo comportamento si osserva anche in esperimenti reali con il batterio e viene ottenuto anche con le simulazioni tramite equazioni differenziali (si veda [17]).

Per applicare a questo modello le strategie di retroazione dall'uscita elaborate nel corso della tesi, è ovviamente necessario stabilire quali siano le uscite prelevabili dal sistema. Sono state effettuate simulazioni utilizzando due diverse uscite:

- caso A: $y_1(t) = M^v(t) \ltimes L_m^v(t)$;
- caso B: $y_2(t) = M^v(t) \ltimes L_m^v(t) \ltimes P^v(t)$.

La scelta di queste variabili di uscita non è obbligata come nel caso della scelta degli ingressi della rete, per cui risulta in qualche misura soggettiva. Nel caso A si è scelto di prendere come uscita solo quelle due variabili perchè fra tutte si ritiene che siano abbastanza facili da misurare e anche abbastanza significative; come si vedrà in seguito, con questa uscita si può fare una retroazione dall'uscita stabilizzante ad uno degli stati 360, 38 e 356, mentre non è possibile stabilizzare la rete allo stato 81. Per poter effettuare retroazione dall'uscita anche rispetto allo stato 81, è stata usata l'uscita y_2, che permette di trovare una retroazione stabilizzante a quello stato. In Tabella 5.2 si riportano i valori di uscita dei punti fissi.

Per quanto riguarda il **caso A**, è stata cercata una retroazione stabilizzante per tutti i quattro possibili punti fissi del sistema; nel seguito si riportano i risultati trovati nei diversi casi:

- stabilizzazione allo stato 360: con la retroazione dall'uscita tempo invariante minima sono state ottenute 81 possibili matrici di

Stato (f v)	y_1 (f v)	y_1 (f l)	y_2 (f v)	y_2 (f l)
360	4	00	8	000
38	1	11	1	111
81	1	11	1	111
356	4	00	8	000

Tabella 5.2: Valori dell'uscita in corrispondenza dei punti fissi della rete.

retroazione dall'uscita, di cui nel seguito ne viene riportata una:

$$K = \left[\begin{array}{cccc} 1 & 1 & 1 & 1 \end{array} \right] .$$

Le altre matrici sono formate da tutte le possibili combinazioni dei primi 3 ingressi (ed infatti sono $81 = 3^4$). Questo implica che in presenza di glucosio extracellulare, la cellula disattiva la trascrizione dell'operone *lac* nel minor numero di passi possibili, qualunque sia la concentrazione del lattosio esterna.
Sono poi state utilizzate le tecniche elaborate per la retroazione dall'uscita tempo invariante più generali, e nello specifico sono state usate la tecnica a forza bruta e il metodo della matrice polinomiale (la tecnica dei cicli diventa invece inapplicabile, perchè l'algoritmo per la ricerca dei cicli è decisamente inefficiente nel caso la rete abbia un numero di nodi abbastanza consistente); i risultati ottenuti, che ovviamente coincidono, forniscono 426 matrici di retroazione dall'uscita distinte, tra cui ci sono tutte le 81 previste con il primo metodo. Nel seguito viene riportata una delle matrici di retroazione che non appartiene a quelle fornite dal metodo di retroazione minima:

$$K = \delta_6 \left[\begin{array}{cccc} 1 & 1 & 4 & 1 \end{array} \right] .$$

Tutte le diverse matrici presentano il primo elemento diverso da δ_6^4 e δ_6^5 (dal momento che con l'uso del primo valore di ingresso lo stato 38 è un punto di equilibrio, mentre con il secondo valore è lo stato 81 a esserlo), e presentano l'ultimo elemento pari a δ_6^1, δ_6^2 o δ_6^3, ingressi che mantengono lo stato 360 in se stesso. Tutti gli altri elementi della matrice di retroazione non hanno alcun vincolo.
È infine stata usata la tecnica di retroazione dall'uscita tempo variante, che in realtà ha fornito come uscita una sequenza di 3 matrici identiche e pari a

$$K(0) = K(1) = K(2) = \left[\begin{array}{cccc} 1 & 1 & 1 & 1 \end{array} \right] ;$$

la soluzione fornita è quindi tempo invariante, e per di più coincide con una soluzione ottenibile con il primo metodo. Nella Tabella 5.3

si riporta l'evoluzione di tutti i nodi della rete usando la matrice ottenuta dalla retroazione tempo variante, in modo da avere un'idea di come la rete evolva nel tempo (il simbolo − indica che la variabile può assumere uno qualsiasi dei due valori possibili).

Variabile (f l)	t=0	t=1	t=2	t=3
M	-	-	0	0
B	-	-	-	0
R	-	-	-	1
A	-	-	0	0
L	-	0	0	0
P	-	-	-	0
C	-	0	0	0
R_m	-	-	-	1
A_m	-	-	0	0
L_m	-	0	0	0

Tabella 5.3: Evoluzione dei nodi della rete Booleana che rappresenta il funzionamento dell'operone *lac* usando la matrice di retroazione dall'uscita tempo variante.

Come si nota, in 3 passi il sistema si porta ad uno stato che è il punto fisso desiderato; da qui, applicando uno dei primi 3 ingressi, la rete resta indefinitamente in quello stesso stato.

- stabilizzazione allo stato 38: utilizzando l'uscita y_1, l'algoritmo per la ricerca di una retroazione tempo invariante minima fornisce una unica matrice:
$$K = \delta_6 \left[\begin{array}{cccc} 4 & 4 & 4 & 4 \end{array} \right] .$$
Visto che lo stato 38 implica la trascrizione dell'operone, si può concludere che per fare in modo che il lattosio sia digeribile dal batterio è sufficiente eliminare dall'ambiente extracellulare il glucosio, ed aggiungere in alta concentrazione il glucosio; in questo modo il batterio comincia nel minor numero possibile di passi a digerire il disaccaride.
L'algoritmo che ricerca la retroazione tempo invariante più generale, fornisce 6 soluzioni distinte: una di queste è la K trovata con il metodo minimo, mentre le altre 5 coincidono con questa a parte per il secondo elemento che può assumere uno qualsiasi dei valori possibili dell'ingresso (e da questo fatto si conclude anche a posteriori che nessun punto fisso della rete fornisce come valore

dell'uscita δ_4^2). Se quindi all'interno della cellula è presente l'mRNA trascritto a partire dall'operone, ma non è presente lattosio, con un qualsiasi tipo di combinazione di zuccheri extracellulari si può comunque mantenere operativo l'operone, ma solo usando l'ingresso δ_6^4 si arriva al punto stabile (e quindi operone trascritto e digestione attiva, dal momento che è presente anche il lattosio) nel minor numero di passi. La tecnica della retroazione tempo variante fornisce la seguente sequenza di 7 matrici di retroazione:

$$K(0) = \delta_6 \begin{bmatrix} 4 & 4 & 4 & 4 \end{bmatrix}; \quad K(1) = \delta_6 \begin{bmatrix} 4 & 1 & 4 & 1 \end{bmatrix};$$
$$K(2) = \delta_6 \begin{bmatrix} 4 & 1 & 4 & 1 \end{bmatrix}; \quad K(3) = \delta_6 \begin{bmatrix} 4 & 1 & 4 & 1 \end{bmatrix};$$
$$K(4) = \delta_6 \begin{bmatrix} 4 & 1 & 4 & 1 \end{bmatrix}; \quad K(5) = \delta_6 \begin{bmatrix} 4 & 1 & 1 & 1 \end{bmatrix};$$
$$K(6) = \delta_6 \begin{bmatrix} 4 & 1 & 1 & 1 \end{bmatrix}; \quad K(7) = \delta_6 \begin{bmatrix} 4 & 1 & 1 & 1 \end{bmatrix}.$$

- stabilizzazione allo stato 81: il tentativo di stabilizzare il sistema a questo stato fallisce utilizzando tutti e tre i diversi metodi elaborati. Come verrà successivamente mostrato, usando l'uscita y_2 sarà possibile stabilizzare il sistema allo stato 81, per cui se fosse necessaria la stabilizzazione a questo stato, bisogna utilizzare un'uscita diversa da y_1, come per l'appunto y_2.

- Stabilizzazione allo stato 356: il metodo di retroazione dall'uscita minima in questo caso non è in grado di fornire alcuna soluzione al problema posto, mentre usando l'algoritmo per la retroazione tempo invariante dall'uscita più generale, il problema può essere risolto attraverso l'uso di 251 diverse matrici di retroazione, di cui se ne riporta una a titolo di esempio:

$$K = \delta_6 \begin{bmatrix} 1 & 1 & 1 & 5 \end{bmatrix}.$$

Analizzando le restanti matrici, si nota che il quarto valore può essere δ_6^5 oppure δ_6^6, mentre per le restanti colonne è presente solo il vincolo che se l'ingresso non possa essere di equilibrio per un qualche stato (che fornisce quella data uscita). È interessante notare che nonostante all'esterno della cellula sia presente del lattosio, la sua concentrazione risulta comunque troppo bassa per innestare tutto il funzionamento dell'operone; probabilmente la quantità di lattosio in grado di penetrare nella cellula per diffusione o grazie alla bassa presenza di permeasi, è troppo bassa per indurre una produzione di allolattosio in grado di legarsi ai repressori e disattivarli, per cui la situazione resta bloccata in una situazione di stallo. Se

successivamente viene aggiunto maggiore lattosio al di fuori della cellula però il processo si innesta in maniera veloce, anche perchè l'attivatore CAP è già attivo (fatto che si comprende osservando le matrici di retroazione per stabilizzare la rete allo stato 38, poiché quando l'uscita è δ_4^4 è sufficiente fornire alla cellula lattosio in gran quantità per indurre la cellula a trascrivere l'operone).
Applicando invece il metodo per ottenere la retroazione tempo variante, si ottiene la seguente sequenza di 7 matrici di retroazione:

$$K(0) = \delta_6 \left[\begin{array}{cccc} 1 & 1 & 1 & 6 \end{array} \right]; \quad K(1) = \delta_6 \left[\begin{array}{cccc} 1 & 1 & 1 & 5 \end{array} \right];$$

$$K(2) = \delta_6 \left[\begin{array}{cccc} 1 & 1 & 1 & 5 \end{array} \right]; \quad K(3) = \delta_6 \left[\begin{array}{cccc} 1 & 1 & 1 & 6 \end{array} \right];$$

$$K(4) = \delta_6 \left[\begin{array}{cccc} 1 & 1 & 1 & 5 \end{array} \right]; \quad K(5) = \delta_6 \left[\begin{array}{cccc} 1 & 1 & 1 & 5 \end{array} \right];$$

$$K(6) = \delta_6 \left[\begin{array}{cccc} 1 & 1 & 1 & 5 \end{array} \right].$$

Se negli altri casi il fatto che la colonna i-esima della matrice passa da un valore δ_6^j, $2 \leq j \leq 6$ in $K(t)$ al valore δ_6^1 in $K(t+1)$ può essere semplicemente sintomo che al tempo $t+1$ nessuno stato in cui si può trovare la rete fornisce come uscita δ_4^i (e questo succede per come è stato elaborato l'algoritmo di ricerca della retroazione tempo variante), in questo caso l'ultima colonna passa da δ_6^5 a δ_6^6 e quindi è certamente un controllo tempo variante.
Questo caso risulta inoltre interessante poiché è evidente che la retroazione dall'uscita minima tempo invariante non è in grado di individuare l'esistenza di una soluzione, anche se effettivamente esistono delle soluzioni al problema tempo invarianti.

Nel seguito si riportano i risultati ottenuti nel **caso B**, cioè usando come uscita $y_2(t) = M^v(t) \ltimes L_m^v(t) \ltimes P^v(t)$, per quanto riguarda il problema della stabilizzazione allo stato 81. Utilizzando il metodo di retroazione dall'uscita minima, l'uso dell'uscita y_2 permette di individuare 8 distinte soluzioni al problema, di cui se ne riporta una:

$$K = \delta_6 \left[\begin{array}{cccccccc} 5 & 4 & 4 & 4 & 4 & 4 & 4 & 4 \end{array} \right];$$

ovviamente la matrice K in questo caso presenta 8 colonne, perché l'uscita y_2 ha dimensione 8. Anche applicando uno degli algoritmi elaborati nella sezione 4.2.2, il problema risulta risolvibile, e in particolare si individuano 432 soluzioni diverse. In ogni caso la prima colonna presenta sempre l'ingresso di equilibrio per lo stato 81, mentre nelle altre colonne oltre ai vincoli imposti all'ottava (che può presentare solo il valore δ_6^4) non tutte le combinazioni sono possibili. È una situazione abbastanza

particolare (e difatti in evoluzione libera con l'ingresso δ_6^5 sono pochissimi gli stati che vengono assorbiti da questo punto fisso), poichè in questa situazione la permeasi è presente, quindi il lattosio esterno, presente in concentrazione media riesce ad entrare nella cellula, ma il livello di lattosio intracellulare resta comunque medio, per cui anche l'allolattosio è in concentrazione media; il repressore continua a essere inattivato dall'allolattosio presente, ma è un equilibrio delicato.
Una possibile sequenza di 6 matrici che permettono la retroazione dall'uscita (di tipo tempo variante) è infine la seguente:

$$K(0) = \begin{bmatrix} 5 & 4 & 4 & 4 & 4 & 4 & 4 & 4 \end{bmatrix};$$

$$K(1) = \begin{bmatrix} 5 & 4 & 1 & 1 & 4 & 4 & 1 & 1 \end{bmatrix};$$

$$K(2) = \begin{bmatrix} 5 & 4 & 1 & 1 & 4 & 4 & 1 & 1 \end{bmatrix};$$

$$K(3) = \begin{bmatrix} 5 & 4 & 1 & 1 & 4 & 4 & 1 & 1 \end{bmatrix};$$

$$K(4) = \begin{bmatrix} 5 & 4 & 1 & 1 & 1 & 4 & 1 & 1 \end{bmatrix};$$

$$K(5) = \begin{bmatrix} 5 & 4 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

In sei passi si è quindi in grado di assicurare il raggiungimento dello stato 81, e per mantenerlo è sufficiente continuare ad applicare l'ingresso δ_6^5, cioè mantenere nell'ambiente circostante al batterio una concentrazione media del lattosio.

Applicando i metodi elaborati per le strategie di controllo dall'uscita sono state individuate diverse matrici di retroazione che permettono di stabilizzare il sistema ad ognuno dei diversi stati che sono per qualche ingresso dei punti fissi. È stato anche visto che con l'uscita y_1 non è possibile stabilizzare il sistema allo stato 81, cosa possibile sfruttando invece l'uscita y_2.
Di certo l'uso di una rete Booleana per modellare il sistema dell'operone *lac* fornisce dei buoni risultati, e le previsioni che si possono fare sfruttando il modello hanno un buon riscontro a livello sperimentale; inoltre questo modello non necessita di tutto un insieme di parametri che invece risulta fondamentale se il processo è modellato tramite equazioni differenziali (in [17], il modello contiene 3 equazioni differenziali, 15 equazioni di aggiornamento e 26 parametri che vanno stimati da dati sperimentali); se si vuole quindi ottenere la sequenza con cui le sostanze sono presenti o assenti, attive o inattive, il modello tramite rete Booleana richiede molti meno dati rispetto al modello tramite equazioni differenziali, viceversa, se si vogliono conoscere anche gli istanti temporali in cui le sostanze si

modificano, le equazioni differenziali restano il metodo migliore, anche se esistono alcuni modelli con reti Booleane che inseriscono anche delle informazioni sulla tempistica.

6 Conclusione

In questo lavoro di tesi sono state trattate le reti Booleane (BN) e le reti Booleane di controllo (BCN), utilizzando l'approcio in forma di stato introdotto in [1]. Per le BN ci si è concentrati sul comportamento a regime, e quindi sui cicli della rete e sui bacini di attrazione di ogni singolo ciclo, ottenendo anche una condizione per la stabilità della rete. Per quanto riguarda le BCN, sono state studiate le diverse proprietà, quindi raggiungibilità e controllabilità, osservabilità e ricostruibilità. È stato anche implementato un algoritmo per la ricerca dei cicli di una rete Booleana di controllo, anche se la sua efficienza è ridotta. Importante in questo ambito è stato lo studio della stabilizzabilità della rete, proprietà che è risultata fondamentale per l'analisi delle tecniche di controllo della rete stessa.
Per quanto riguarda il controllo di una BCN, la prima tecnica studiata è stata la retroazione dallo stato; per applicare questo metodo è sufficiente che la rete sia stabilizzabile ad uno stato, e l'ingresso può essere calcolato attraverso una matrice di retroazione, cioè nella forma $u(t) = Kx(t)$. Si è successivamente passati alla retroazione dall'uscita, riportando l'unico studio presente in letteratura, [6], evidenziandone però una criticità; sono state quindi elaborate delle nuove tecniche di retroazione dall'uscita, sia di tipo tempo invariante che di tipo tempo variante. Come è stato evidenziato, se la soluzione esiste per la retroazione dall'uscita minima, allora esiste di certo una soluzione per la retroazione tempo invariante introdotta in questa tesi, e a sua volta esiste una retroazione dall'uscita tempo variante. D'altro canto l'esistenza di una retroazione tempo variante non assicura quella di una retroazione tempo invariante, per cui la retroazione tempo variante risulta il metodo più generale che è stato introdotto in questa tesi. Ultima strategia di controllo riportata è il controllo ottimo, di cui si sono riportate diverse tipologie studiate in [7].
Nell'ultimo Capitolo le strategie di controllo tramite retroazione dall'uscita sono state applicate ad un caso pratico costruito a partire da un sistema biologico: il funzionamento dell'operone *lac*. Si è inizialmente studiato il funzionamento a livello biologico ed è stato costruito il relativo modello matematico; dopo aver trovato la forma algebrica sono stati confrontati i risultati ottenuti con quelli presentati nell'articolo da cui è stato tratto il modello, [12], ottenendo una totale corrispondenza. Sono state infine applicate le tecniche di retroazione dall'uscita elaborate nel corso della tesi.

Come sviluppi futuri, l'aspetto più importante sarebbe quello di tro-

vare una condizione necessaria e sufficiente per l'esistenza di una retroazione dall'uscita tempo variante; la condizione fornita in questo lavoro risulta solo sufficiente, ed in effetti esistono delle reti che non rispettano la condizione del teorema trovato e, nonostante questo, sono controllabili tramite una retroazione dall'uscita.
Potrebbe poi essere utile elaborare un algoritmo per la ricerca dei cicli di una BCN che abbia una efficienza maggiore.

A Funzioni Matlab

A.1 Individuazione dei cicli

La prima funzione Matlab qui riportata fornisce tutti i cicli della BCN in esame; come è già stato sottolineato nel Capitolo 3 è una funzione che può essere applicata solo se le dimensioni della rete sono contenute.

```
%% la funzione individua tutti i cicli (semplici) di una BCN

% input: L, matrice di evoluzione della BCN;
%        P, numero di ingressi della rete

%output: x, vettore che contiene tutti i cicli della rete (ogni ciclo \'e
%   scritto come una successione di coppie stato-ingresso);
%   y, contiene gli indici di partenza dei diversi cicli

function [g1 g2]=cicliBCN(L,P)

N=length(L)/P;   %numero di stati della rete
global x;
global y;
x=[]; %conterr\'a tutti i cicli della rete, scritti come (stato-ingr),
y=[1]; %e gli indici di inizio di cicli diversi sono contenuti in y
for i=1:N    %a partire da ogni stato cerco i cicli che hanno
    z=i;     %z come stato iniziale
    t=[];
    trovacicliBCN(z,t,L,P,i);
end
g1=x;
g2=y;
end

%% Funzione ricorsiva che stabilisce quali sono i cicli che presentano
% numSt come stato iniziale

%input: z, stati esaminati della rete
%       t, contiene gli ingressi che permettono di passare da numSt a z(end)
%       L, matrice di transizione della BCN
%       P, numero di ingressi della rete
%       numSt, stato iniziale del ciclo

function trovacicliBCN(z,t,L,P,numSt)
```

```
global x;     %contiene coppie stato−ingresso dei cicli
global y;     %contiene indici di x per i cicli
N=length(L)/P;   %dimensione stato
sa=z(end);   %stato in cui \'e il sistema
a=[];     %stati in cui si finisce da sa
for i=1:P
    a=[a L((i-1)*N+sa)];
end
p=find(a>=numSt);     %se si arriva in uno stato 'i' che \'e minore di numSt,
                      %l'eventuale ciclo che si crea \'e gi\'a stato
                      %inserito in x, perch\'e gi\'a stato inserito durante la
                      %ricerca dei cicli che coinvolgono lo stato 'i'
a=a(p);       %stati dove posso finire esclusi quelli di cui ho gi\'a trovato i cicli
c=[1:P];
c=c(p);       %ingressi che permettono il passaggio da sa a a (el i−esimo di c
              %fa passare da sa a elemento i−esimo di a)
r=find(a==numSt);     %se qualche elemento di a coincide con numSt, allora \'e
                      %stato trovato un ciclo
s=[];     %indici per cui dopo chiamare ricorsione
for i=1:length(a)     %per tutti gli elementi di a
    check=1;          %se diventa 0, o \'e ciclo cercato (viene trattato a parte)
                      %o non sar\'a un ciclo semplice ed \'e quindi inutile
                      %applicare la ricorsione
    for j=1:length(z)
        if a(i)==z(j)     %stato a(i) \'e gi\'a stato esplorato in precedenza
            check=0;      %questa sequenza non interessa
        end
    end
    if check==1 %tutti gli stati sono distinti, si pu\'o utilizzare ulteriormente funzione
        s=[s i];
    end
end
if(isempty(r)==0)     %elementi per cui si forma ciclo
    for j=1:length(r)
        c1=[];   %conterr\'a nuovo ciclo da aggiungere
        for h=1:length(t)
            c1=[c1 z(h) t(h)];
        end
        c1=[c1 z(end) c(r(j))];       %inserito anche ultimo stato−ingresso
        x=[x c1];         %aggiungo ciclo trovato
        y=[y length(x)+1];   %aggiorno indice
    end
end
```

```
if (isempty(s)==0)
    for i=1:length(s)
        z=[z a(s(i))];   %viene aggiunto il nuovo stato su cui si arriva e
        t=[t c(s(i))];   %ingresso per raggiungerlo
        trovacicliBCN(z,t,L,P,numSt)      %viene richiamata la funzione con z
                                          %e t aggiornati
        z=z(1:end-1);     %se s ha pi di un elemento, allora \'e necessario
                          %che z e t tornino quelli
        t=t(1:end-1);     %di partenza per riapplicare funzione
    end
end
end
```

A.2 Retroazione dall'uscita minima tempo invariante

La seguente funzione individua la matrice di retroazione dall'uscita minima, e, come è già stato sottolineato in 4.2.1, è possibile sfruttarla per trovare anche una matrice di retroazione minima dallo stato.

```
%% trova la retroazione dall'uscita che si basa sulla retroazione minima
% dallo stato, \'e funzione legata all'articolo [4]

%input: vL matrice di transizione della BCN in forma vettoriale; H matrice
%   stato-uscita, forma vettoriale; N dimensione stati; Ping dimensioni
%   uscite(si suppone che tutte possano uscire, altrimenti si rinumerano
%   uscite e quelle che non compaiono sono le ultime),
%   M dimensione ingressi, e stato a cui si vuole stabilizzare il sistema.

%output: K la matrice di retroazione dall'uscita, pu\'o essere pi di una e
%   gli indici dell'inizio delle singole matr sono contenuti in indK.

function [K indK]=retArtOutFeed(vL,H,N,Ping,M,e)
%controlla che BCN sia stabilizzabile a e
L=creaLBCN(vL,N);     %matrice L in forma estesa
Ltot=L(:,1:N);   %creazione di Ltot
for i=1:M-1    %per ogni ingresso aggiungo elementi a Ltot
    Ltot=Ltot|L(:,i*N+1:(i+1)*N);
end
st=[1:N];
corone=zeros(1,N);   %nella pos i conterr\'a la corona a cui appartiene lo
```

```
                        %stato delta_N^i
if Ltot(e,e)==0 %lo stato e non \'e di equilibrio=>errore
    disp('lo␣stato␣a␣cui␣si␣vuole␣stabilizzare␣non␣\'e␣di␣equilibrio
␣␣␣␣␣␣␣␣per␣nessun␣ingresso');
    K=[];
    indK=[];
    return
else
    st(e)=0;
end
Ltot=double(Ltot);
Ltotpot=Ltot;
for i=1:N-1    %calcola potenze di Ltot per stabilire se \'e stabilizzabile
                % e divide stati in corone
    for j=1:N    %guarda le corone
        if Ltotpot(e,j)~=0 && st(j)~=0    %\'e ancora pot i-1, se el \'e diverso
            st(j)=0;          %da 0 e st rel \'e 0, allora \'e prima volta che
            corone(j)=i;      %stato j ragg e e quindi appartiene a corona i
        end
    end
    Ltotpot=Ltotpot*Ltot;
    Ltotpot=Ltotpot>0; %non mi importa valore, basta che sia diverso da 0
    Ltotpot=double(Ltotpot);
end
if isempty(find(Ltotpot(e,:)==0))==0   %non \'e raggiungibile da tutti gli
                                        %stati=>no stabilizzabilit\'a
    disp('lo␣stato␣a␣cui␣si␣vuole␣stabilizzare␣non␣\'e␣raggiungibile
␣␣␣␣␣␣␣␣da␣tutti␣gli␣stati');
    K=[];
    indK=[];
    return
end
P=[];    %conterr\'a i Pi relativi ad ogni stato
indP=[1];      %el i contiene indice di P da cui parte Pi dello stato i
for i=1:N    %costruzione dei Pi
    cor=corone(i);   %corona a cui appartiene stato in esame
    if cor==0    %\'e stato di equilibrio
        for j=1:M %cerco quali sono ingr di equil
            if vL((j-1)*N+i)==i       %\'e ingr di equil
                P=[P j];
            end
        end
    else      %non \'e stato a cui bisogna stabilizzare
        for j=1:M %cerco quali sono ingr che portano a corona minore
```

```
            if corone(vL((j-1)*N+i))==cor-1 %se stato a cui arrivo
                                %appartiene a corona minore lo aggiungo
                P=[P j];
            end
        end
    end
    indP=[indP length(P)+1];      %aggiornamento indice P
end
O=zeros(1,Ping);    %conterr\'a gli Ok relativi ad ogni uscita
indO=[1:Ping+1];    %el i contiene indice di O da cui parte Ok
for i=1:N    %inserisco ogni stato nel relativo Ok
    kk=H(i);      %Ok a cui appartiene stato in esame
    if O(indO(kk))==0
        O(indO(kk))=i;
    else
        O=[O(1:indO(kk+1)-1) i O(indO(kk+1):end)];   %viene inserito alla
        indO(kk+1:end)=indO(kk+1:end)+1;      % fine degli Ok fin li trovati
    end
end
I=zeros(1,M*Ping); %contiene per ogni possibile uscita M elementi,
                   %se el i-esimo \'e diverso da zero allora i \'e ingr
                %che va bene a tutti gli ingressi dell'uscita considerata
for i=1:Ping     %si controllano tutti gli Ok
    stesam=O(indO(i):indO(i+1)-1);   %stati che appartengono a Oi
    stes=stesam(1); %primo stati di quelli
    for j=1:indP(stes+1)-indP(stes) %analizza tutti gli ingressi che vanno
                                    %bene per stes
        ingr=P(indP(stes)+j-1); %ingresso in analisi
        check=1;
        for h=2:length(stesam)   %per ogni ingr che va bene a stes bisogna
                                 %controllare che vada bene per altri
            stes2=stesam(h);
            if check==1    %ingr va bene fino allo stato h-1-esimo di stesam
                check=0; %diventa 1 se c'\'e stato h-esimo con stesso ingresso
                for l=1:indP(stes2+1)-indP(stes2)     %si controlla ogni
                                                     %ingresso relativo a stes2
                    ingr2=P(indP(stes2)+l-1);
                    if ingr==ingr2
                        check=1;   %ingresso di stes va bene anche per stes2
                    end
                end
            end
        end
        if check==1       %ingresso ingr va bene per tutti gli stati
```

```
            I((i-1)*M+ingr)=ingr;
        end
    end
end
for i=1:Ping    %se un dato Ik \'e tutto 0 non esiste retroazione
    if I((i-1)*M+1:i*M)==zeros(1,M)
        disp('non␣esiste␣retroazione␣dall''uscita␣con␣il␣metodo␣cercato.');
         K=[];
        indK=[];
        return;
    end
end
K=[]; %conterr\'a tutte le possibili matr di retroazione
indK=[1];    %usato e gli indici alle diverse matr sono contenuti in indK
for h=1:M    %ingressi relativi a uscita 1
     if I(h)~=0      %se el i dei primi M \'e diverso va inserito
        K=[K I(h)];
        indK=[indK length(K)+1];
     end
end
    for j=2:Ping    %aggiungo tutti i rimanenti ingressi
        inges=I(M*(j-1)+1:j*M); %ingressi che vanno bene per stati
                                %che forniscono come uscita j
        a=sum(inges>0);   %numero di ingressi che vanno bene
        if a>1 %ho pi di un ingresso=> per ogni possibile gruppo di
              %ingressi che vanno bene per le uscite 1,...,j-1 si devono
              %aggiungere 'a' gruppi, ognuno per ogni ingresso distinto
              %che si pu\'o dare quando l'uscita \'e j
            hh=K;   %ingresso di retroazione per uscite 1,...,j-1
            indhh=indK; %relativi indici
            K=[];   %azzerato
            indK=[];
            for h=1:M    %si analizzano tutti i possibili ingressi da
                         %associare all'uscita j
                aa=hh;
                bb=indhh;
                if inges(h)~=0 %ingresso \'e accettabile
                    for l=1:length(bb)-1 %per ogni gruppo di ingr contenuto
                                         %aggiungo l'ingr per l'uscita j
                        aa=[aa(1:bb(l+1)-1) inges(h) aa(bb(l+1):end)];
                        bb(l+1:end)=bb(l+1:end)+1;
                    end
                     K=[K aa]; %avendo aggiunto l'ingr per tutti i gruppi
                               %di ingressi si aggiorna K
```

```
                    if isempty(indK)     %aggiornamento degli indici
                        indK=[bb];
                    else
                        indK=[indK bb(1:end-1)+(indK(end)+j-1)];
                    end
                end %si ripete questa operazione finch\'e non sono finiti
                    %i possibili ingressi per l'uscita j
            end
        else %c'\'e solo un ingr che va bene per l'uscita j=>basta
                %aggiungerlo ad ogni gruppo di ingressi
            for h=1:M
                if inges(h)~=0
                    for l=1:length(indK)-1
                        K=[K(1:indK(l+1)-1) inges(h) K(indK(l+1):end)];
                        indK(l+1:end)=indK(l+1:end)+1;
                    end
                end
            end
        end
    end
end
```

A.3 Retroazione dall'uscita tempo invariante

In questa sezione si riportano i tre diversi algoritmi elaborati nella Sezione 4.2.2.
Per prima si riporta la tecnica tramite matrice polinomiale.

```
%% La funzione calcola la matrice polinomiale Lcorsivo e la sua potenza
% n-esima (non considerando per\'o i prodotti misti tra ingressi relativi a
% una stessa uscita che sono diversi)

%input: vL \'e la matrice L in versione vettoriale; n \'e il numero di stati
%   della BN; b  un vettore che contiene informazioni rispetto all'uscita,
%   in particolare se i primi i stati danno l'uscita 1, i secondi j danno 2
%   e cos via b=[i i+j ...]


%output: PA2 contiene la potenza n-esima di Lcorsivo; A contiene Lcorsivo

%Lcorsivo e le sue potenze sono riportate nel seguente modo: se l'elemento
%\'e composto da i monomi, il primo elemento di Lcorsivo \'e i. Poi vengono
```

```
%inseriti i monomi uno dietro l'altro, con la seguente notazione: se un
%monomio \'e composto da j diverse incognite, e l'incognita z_{ni_h,h} \'e
%elevata alla potenza ph, allora ogni monomio \'e cos rappresentato:
%[j ni_1 1 p1 ... ni_j j pj]. In definitiva quindi un elemento di questa
%matrice \'e del tipo
%[i j1 ni_11 11 p11 ... ni_j1 j1 pj1 ... ji ni_1i 1i p1i ... ni_ji ji pji]

function [PA2,A]=metodoPolinomiale(vL,n,b)
r=length(vL)/n; %numero di ingressi
A=cell(n,n);      %conterr\'a matrice Lcorsivo
for i=0:r-1 %per ogni ingresso
    p=1;      %contiene a quale gruppo colonna di uscita appartiene
              %l'indeterminata
    for j=1:n    %per tutti gli stati
        if j>b(p)    %lo stato non crea pi ingresso p ma quello successivo
            p=p+1;
        end
        m=j+n*i; %se sono nello stato j, con l'ingresso i bisogna guardare
                 %questo indice di colonna in L per capire stato dove si va
        if isempty(A{vL(m),j}) %non c'\'e ancora alcun elemento=>si inserisce
                               %il primo
            A{vL(m),j}=[1 1 i+1 p 1];
        else    %si inserisce un altro monomio
            A{vL(m),j}=[A{vL(m),j} 1 i+1 p 1];
            A{vL(m),j}(1,1)=A{vL(m),j}(1,1)+1;   %si aggiorna il numero di
                                                 %monomi contenuti
        end
    end
end
PA1=A;  %corrisponde a Lcorsivo
for i=1:n-1    %elevazione alla 2^n-esima pot
    PA2=cell(n,n);
    for j=1:n  %colonna di PA2/ colonna da cui prelevare da A
        for k=1:n    %indice di riga col j di A
            if (isempty(A{k,j})==0)
                for l=1:n  %riga su PA2 (col j)
                    if (isempty(PA1{l,k})==0)   %c'\'e la "moltiplicazione
                                                %da effettuare"
                        a=A{k,j};%elemento di A con cui si effettua moltipl
                        pa=PA1{l,k};   %elemento di PA1 da moltiplicare
                        blEsA=2; %indice dell'inizio del primo monomio di a
                        numAddA=a(1);     %num monomi di a
                        numAddPA=pa(1); %num monomi di pa
                        for h1=1:numAddA      %per ogni monomio di a
```

```
ai=a(blEsA+1:blEsA+3);   %monomio in esame di a
        %(sono tutti monomi di grado 1 in a)
blEsPA=2;     %indice dell'inizio del primo
                    %monomio di pa
pot=0;        %se va a uno vuol dire che \'e stata
   %fatta la potenza di un monomio gi\'a presente
for h2=1:numAddPA     %per ogni monomio di pa
    check=1;
    for t=1:pa(blEsPA)
            %analizzo che se monomio ai
            %appartiene a stesso blocco colonna
            %di una delle indeterminate di pa,
            %allora abbia stesso ingresso,
            %altrimenti check a 0
        if ((ai(2)==pa(blEsPA+(t-1)*3+2))&&...
             ...(ai(1)~=pa(blEsPA+(t-1)*3+1)))
            check=0;
        else if ((ai(2)==pa(blEsPA+(t-1)*3+2))...
            ...&&(ai(1)==pa(blEsPA+(t-1)*3+1)))
                    %se ha lo stesso ingresso
                %bisogna aggiungere una potenza
                pot=1;
                tpot=t;  %indeterminata a cui
                        %aggiungere potenza
            end
        end
    end
    if check==1  %monomio ai \'e compatibile con
            %resto di monomio di pa in esame
        x=pa((blEsPA+1):blEsPA+pa(blEsPA)*3);
        blEs2=pa(blEsPA);
        if pot==1%se c'era una potenza maggiore
            x((tpot-1)*3+3)=x((tpot-1)*3+3)+1;
      %si aumenta la potenza dell'indeterminata
            pot=0;
        else      %monomio di ai va aggiunto
            blEs2=blEs2+1;   %si aggiunge nuova
            %indeterminata e quindi va contata
            x=[x ai];
        end
        pa2=[blEs2 x];   %elemento che va
      %aggiunto sulla matrice potenza in pos l j
        if isempty(PA2{l,j})      %aggiornamento
                                %se el l j \'e vuoto
```

```
                                        PA2{l,j}=[1 pa2];
                                    else                        %aggiornamento
                                                                %se el l j \'e gi\'a pieno
                                        PA2{l,j}=[PA2{l,j} pa2];
                                        PA2{l,j}(1,1)=PA2{l,j}(1,1)+1;
                                    end
                                end
                                blEsPA=blEsPA+pa(blEsPA)*3+1 %aggiornamento
                                        %per passare al monomio succ di pa
                            end
                             blEsA=blEsA+3+1;    %aggiornamento per passare
                                                 %al monomio successivo di a
                        end
                    end
                end
            end
        end
    end
    PA1=PA2;      %la nuova matrice trovata viene usata al passo
                  %successivo come la PA1 di partenza
end
end

%% la funzione prende come ingressi la matrice polinomiale N (potenza
% n-esima di Lcorsivo e la analizza per stabilire se esistono e quali siano
% eventualemente gli ingressi di retroazione

function [ret m]=stabilire(N,n)
ret=[0];      %conterr\'a i possibili ingressi di retroazione
%primo el di ret \'e quante retroazioni sono possibili
x=N{1,1};     %primo elemento
if isempty(x)     %non esiste ingresso per il quale  xe \'e punto fisso
    disp('Non␣esiste␣la␣retroazione␣dall''uscita')
    return
end
numAdd=x(1);      %quanti monomi in posizione 11
elBl=2;           %quale monomio \'e esaminato
m=[2];            %tiene l'indice di inizio in ret delle diverse retroazioni
for i=1:numAdd
    if x(elBl)==1    %monomio composto da un solo ingresso (\'e punto fisso)
        ret=[ret x(elBl+1) x(elBl+2)];   %inseriamo questo elemento
        ret(1)=ret(1)+1;
        m=[m length(ret)+1];
    end
```

```
    elBl=elBl+x(elBl)*3+1;   %monomio successivo da controllare
end
if ret(1)~=0      %esiste almeno un ingresso per cui xe \'e punto fisso
    for i=2:n     %analizzo tutti i restanti elementi
        retn=[0];%conterr\'a possibili retroazioni dopo aver analizzato N(1,i)
        mn=[2];
        x=N{1,i};     %elemento i-esimo
        if isempty(x)     %non esiste seq di ingressi per cui 1 \'e ragg da i
            disp('Non esiste la retroazione dall''uscita')
            ret=[];
            m=[];
            return
        end
        numret=ret(1);
        for j=1:numret   %si analizzano tutte le possibili retroazioni una
            retes=ret(m(j):m(j+1)-1); %si considera una retroazione
                                      %possibile fino all'el i-1
            numAddx=x(1);     %num di monomi di cui \'e composto el N(1,i)
            elBlx=2;      %monomio in esame
            for k=1:numAddx       %si analizzano singoli monomi di N(1,i)
                y=x(elBlx+1:3:elBlx+x(elBlx)*3); %contiene gli ingressi
                                                 %richiesti
                z=x(elBlx+2:3:elBlx+x(elBlx)*3); %contiene a quale Ti \'e rel
                                                 %ingresso
                check=1;
                prova=[];
                u=[1:length(retes)];      %se \'e diverso da zero relativo el
                                          %di retes non \'e stato analizzato
                v=[1:length(y)];          %se \'e diverso da zero relativo el
                                          %di y o z non \'e stato analizzato
                for h=1:length(retes)/2       %analizza tutti el della
                                              %retraozione possibile fino a i-1
                    for t=1:length(z)       %analizza tutti el del monomio di
                                            %N(1,i) in esame
                        if (retes(h*2)==z(t))&&(retes(h*2-1)==y(t))
                %appartengono a stesso Ti e ingresso \'e uguale => \'e accettabile
                            prova=[prova retes(h*2-1) retes(h*2)];
                            u(h*2-1)=0; %el di retes \'e stato analizzato
                            u(h*2)=0;   %idem
                            v(t)=0;     %el di y e z analizzato
                        else if (retes(h*2)==z(t))&&(retes(h*2-1)~=y(t))
                %uguale Ti vuole diversi ingressi=>non va bene retroazione
                            check=0;
                        end
```

```
                    end
                end
            end
                if check==1      %la retroazione era accettabile
                    u=u(u>0);    %vanno aggiunte le parti di retes
                    v=v(v>0);    %e del monomio che non sono state messe
                                 %prima (tutte relative a Ti diversi e
                                 %quindi di certo compatibili)
                    pp=[];
                    for s=1:length(v)
                        pp=[pp y(v(s)) z(v(s))];
                    end
                    prova=[prova retes(u) pp];
                    retn=[retn prova];  %si aggiunge la retroazione
                                 %che fino all'elemento i \'e accettabile
                    retn(1)=retn(1)+1;
                    mn=[mn length(retn)+1];
                end
            elBlx=elBlx+x(elBlx)*3+1;    %si passa al monomio successivo
                                         %di N(1,i)
        end
    end
    if retn(1)==0    %dopo aver analizzato tutto N(1,i) non esiste
                     %alcuna retraozione compatibile
        disp('non esiste retraozione dall''uscita')
        ret=[];
        m=[];
        return
    else
        ret=retn;
        m=mn;
    end
  end
else     %non c'\'e ingresso tale per cui xe \'e punto fisso
   disp('Non esiste la retroazione dall''uscita')
   ret=[];
   m=[];
   return
end
end
```

Di seguito si riporta la tecnica con l'analisi dei cicli.

```
%%La funzione svolge i seguenti compiti:
% 1 dispone i cicli contenuti in x in ordine di lunghezza (y aggiornata
%   di conseguenza)
% 2 si mantengono solo i cicli relativi al punto fisso e quelli che possono
%   creare problemi nella retroazione statica
% 3 analizza i cicli rimasti (sono quelli che formano Ncorsiva) e trova
%   ingressi di retroazione compatibili con tutti

%input: x cicli BCN; y indici dei diversi cicli di x; N numero stati BCN; P
%       numero uscite BCN; H matrice uscita BCN; M numero ingressi BCN;

%output: s cicli in ordine lunghezza; u indici relativi a s, xn cicli
%   problematici; yn relativi indici; ingok ingressi compatibili per la
%   retroazione contiene vettore multiplo di MP, ogni sottovettore ha per
%   ogni uscita i possibili ingressi compatibili; indingok relativo indice;
%   ingfin contiene vettore multiplo di M, in cui ad ogni sottovettore l'el
%   i \'e quello da associare all'uscita i-esima; indif relativo indice.

function [s u xn yn ingok indingok ingfin indif]=esaminaCicli(x,y,N,P,H,M)
%% 1 disporre cicli in ordine di periodo
s=[];
u=[1];
for i=1:N %cicli al massimo lunghezza N
    for j=1:length(y)-1      %analizza tutti i cicli
        if y(j+1)-y(j)==i*2      %al ciclo i-esimo inserisco solo cicli di
                                  %lunghezza i
            ind=[y(j):y(j+1)-1];
            s=[s x(ind)];
            u=[u length(s)+1];
        end
    end
end
%% selezionare solo cicli che possono creare problemi
x=s;      %cicli in ordine di lunghezza
y=u;
i=1;      %conterr\'a l'el di y che contiene l'indice di x relativo al primo
              %ciclo di lungh>1
check=1;
while (check==1) && (i<=length(y)-1)
    if (y(i+1)-y(i))>2   %il ciclo relativo all'indice succ \'e di lungh>1
        check=0;
    else
        i=i+1;
    end
```

```
end
fpf=i;   %\'e l'indice dell'el di y che contiene l'ind di x da cui parte ciclo
             %lungh>1
xn=x(1:y(i)-1); %contiene tutti i cicli di lungh 1
yn=y(1:i);
ingpf=[];    %conterr\'a gli ingressi che permettono di lasciare xe invariato
for kk=1:2:length(xn)
    if xn(kk)==1
        ingpf=[ingpf xn(kk+1)];
    end
end
for j=i:length(y)-1   %analizza cicli di lungh>1 (j contiene indici rel a y)
    a=[1:P];      %se el i-esimo diventa 0 gi\'a controllato che ing rel a
                  %uscita i sono tutti uguali per il ciclo in esame
    check=1;      %se resta 1 \'e ciclo che pu\'o avvenire con ret uscita
    for h=y(j):2:y(j+1)-3 %h contiene indici di x degli stati del ciclo in
                          %analisi
        if (isempty(find(H(x(h))==a, 1))==0)&&(check==1) %stato esaminato
    %non app a un Ti gi\'a esaminato e \'e possibile che ciclo sia problematico
            a(a==H(x(h)))=0;      %si pone a 0 elemento di a relativo
            %all'uscita corrispondente a x(h)
            for k=h+2:2:y(j+1)-1 %si analizzano i rimanenti stati del ciclo
                if check==1 %solo se ciclo pu\'o essere problematico
                    if ((H(x(h))==H(x(k)))&&(x(h+1)~=x(k+1)))||...
              ((H(x(h))==1)&&(isempty(find(double(x(h+1)==ingpf))>0)))||...
              ((H(x(k))==1)&&(isempty(find(double(x(k+1)==ingpf))>0)))
          %stati appartengono a stesso Ti ma vogliono ingr diversi x(h)
          %appar T1 con ingr diverso da equil per xe
          %x(h) appar T1 con ingr diverso da equil per xe
                        check=0;      %il ciclo non pu\'o crearsi con
                                      %retro statica dall'uscita
                    end
                end
            end
        end
    end
    if check==1 %il ciclo pu\'o crearsi con retro statica
        ind=[y(j):y(j+1)-1];
        xn=[xn x(ind)];       %aggiunto ai cicli problematici
        yn=[yn length(xn)+1];
    end
end
% costruzione ingressi retroazione
pr=[];   %conterr\'a ingressi che sono utilizzabili (non creano cicli
```

```
        %lungh 1 e sono di eq per xe)
        %\'e vettore MP fatto da P vettori [1 2 ... M] in cui el i viene
        %posto a 0 se \'e ingresso che non pu\'o essere usato
inpr=[1];
for i=1:P
    for j=1:M
        pr=[pr j];
    end
    inpr=[inpr length(pr)+1];
end
ingPE=[];    %ingressi che assicurano che xe sia punto fisso
for i=1:fpf-1    %cicli di lungh 1
    if x(y(i))==1    %ciclo di lungh 1 che coinvolge stato xe
        ingPE=[ingPE x(y(i)+1)];
    else     %ciclo non coinvolge xe=>ing rel a suo Ti non pu\'o essere usato
        Tes=H(x(y(i)));
        pr(inpr(Tes)+x(y(i)+1)-1)=0;
    end
end
for i=1:inpr(2)-1    %ingressi relativi a T1
    if isempty(find(pr(i)==ingPE))   %se ingresso non compare in ingPE non
                                      %viene preso (non sarebbe di equil)
        pr(i)=0;
    end
end
global ingok; %conterr\'a ingressi che possono andare bene, pu\'o contenere pi
                %vettori MP. Indici relativi a dove partono diversi vettori
global indingok;     %sono contenuti in indingok
ingok=[];
indingok=[1];
ch=1;
if length(yn)<=fpf   %gli unici cicli da controllare hanno lunghezza 1=>
                        %pr contiene gi\'a ingressi compatibili
    jj=length(pr)/P;     %coincide con M
    for i=1:P    %se per determinato Ti tutti gli ingressi sono a 0 vuol
                 %dire che non esiste retroazione
        if pr((i-1)*jj+1:i*jj)==zeros(1,jj)
            ch=0;
        end
    end
end
if length(yn)>fpf    %esistono cicli di lungh>1 da controllare=>si chiama
                      %funzione esCicRic
    esCicRic(H,xn,yn,fpf,pr,inpr,P);
```

```
elseif ch==1      %solo cicli di lungh 1 ma compatibili
    ingok=pr;
    indingok=[1 length(ingok)+1];
end       %se ch==0 allora ingok \'e rimasto vuoto
if isempty(ingok)==1      %non esiste un ingresso che permetta la retroazione
    disp('Non␣esiste␣retroazione␣statica␣dall''uscita');
    ingfin=[];
    indif=[];
    return;
end
ingfin=[]; %conterr\'a tutti i possibili ingressi di retroazione voluti,
           %sar\'a multiplo di P e sottovettori conterranno in posizione
         % i l'ingresso da dare quando l'uscita \'e i
indif=[];
for i=1:length(indingok)-1   %analizza ogni gruppo di ingressi che
                             %rendono possibile la retroazione
    ingret=[];        %contiene tutti gli ingressi di retroazione
                      %ricavabili da gruppo ingr i
    indir=[1];
    for h=1:M    %ingressi relativi a uscita 1
        if ingok(indingok(i)+h-1)~=0      %se el i dei primi P elementi del
                %gruppo ingr i \'e diverso da 0, allora usando quell'ingr
                % quando l'usc \'e 1 pu\'o esserci retrozione
            ingret=[ingret ingok(indingok(i)+h-1)];
            indir=[indir length(ingret)+1];
        end
    end       %ingret contiene gli ingressi relativi agli stati di T1
    for j=2:P    %analizzo ingressi di ogni altro stato
      inges=ingok(indingok(i)+M*(j-1):indingok(i)+M*j-1);%ingr relativi a j
        a=sum(ingok(indingok(i)+M*(j-1):indingok(i)+M*j-1)>0); %num di ingr
                                              %utilizzabili per uscita j
        if a>1 %ho pi di un ingresso=> per ogni possibile gruppo di
             %ingressi che vanno bene per le uscite 1,...,j-1 devo
             %aggiungere 'a' gruppi, ognuno per ogni ingresso distinto
             %che posso dare quando l'uscita \'e j
            hh=ingret;   %ingresso di retroazione per uscite 1,...,j-1
            indhh=indir; %relativi indici
            ingret=[];   %azzerato
            indir=[];
            for h=1:M    %si analizzano tutti i possibili ingressi da
                         %associare all'uscita j
                aa=hh;
                bb=[indhh];
                if inges(h)~=0 %ingresso \'e accettabile
```

```
                    for l=1:length(bb)-1 %per ogni gruppo di ingr contenuto
                                         %aggiungo l'ingr per l'uscita j
                        aa=[aa(1:bb(l+1)-1) inges(h) aa(bb(l+1):end)];
                        bb(l+1:end)=bb(l+1:end)+1;
                    end
                     ingret=[ingret aa]; %avendo aggiunto l'ingr per tutti
                                         %i gruppi di ingressi si aggiorna ingret
                     if isempty(indir)   %aggiornamento degli indici
                        indir=[bb];
                     else
                        indir=[indir bb(1:end-1)+(indir(end)+j-1)];
                     end
                end %si ripete questa operaizione finch\'e non sono finiti
                    %i possibili ingressi per l'uscita j
            end
        else %c'\'e solo un ingr che va bene per l'uscita j=>basta
             %aggiungerlo ad ogni gruppo di ingressi
            for h=1:M
                if inges(h)~=0
                    for l=1:length(indir)-1
              ingret=[ingret(1:indir(l+1)-1) inges(h) ingret(indir(l+1):end)];
                        indir(l+1:end)=indir(l+1:end)+1;
                    end
                end
            end
        end
    end
    ingfin=[ingfin ingret]; %aggiunto tutti i gruppi di ingresso
                            %relativi al gruppo di ingressi i
                            %si aggiunge quindi a quello definitivo
    if isempty(indif) %aggiornamento indici
        indif=[indir];
    else
        indif=[indif indir(1:end-1)+indif(end)+M-1];
    end
end
%si eliminano eventuali ripetizioni di ingressi di retroazione
fine=indif(end-1); %ultimo gruppo di ingressi di retroazione
i=1;     %si parte dal primo gruppo
while i<=fine    %finch\'e ci sono gruppi da controllare
    inges=ingfin(i:i+P-1);   %gruppo di ingressi con cui si confronta
    j=i+P;   %gruppo con cui confrontare
    while j<=fine    %finch\'e non si raggiunge ultimo gruppo
        if ingfin(j:j+P-1)==inges    %se gruppi uguali
```

```
            ingfin(j:j+P-1)=[];        %il secondo viene tolto
            fine=fine-P;               %indici aggiornati
        end
        j=j+P;   %aggiornato gruppo con cui si confronta
    end
    i=i+P;   %aggiornamento gruppo da controllare
end
indif=[1:P:fine+P];
end

%%funzione ricorsiva indy contiene l'indice di y relativo all'inizio in x
%%del ciclo da analizzare successivamente, pr gli ingressi compatibili fino
%%a quel punto.

function esCicRic(H,x,y,indy,pr,inpr,P)
global ingok;
global indingok;
M=length(pr)/P;
for i=1:P    %se per un dato Ti tutti gli ingressi non sono accettabili
             %non esiste retroazione
    if pr((i-1)*M+1:i*M)==zeros(1,M)
        return;
    end
end
xes=x(y(indy):y(indy+1)-1); %ciclo sotto esame
ingNo=zeros(1,P);          %contiene gli ingressi che attivano il ciclo
                  %(in pos i-esima \'e ingr che attiva ciclo rel a stati di Ti)
for i=1:2:length(xes)
    ingNo(H(xes(i)))=xes(i+1);
end
check=0;      %se diventa uno, la ricorsione si chiama con stesso pr
for i=1:length(ingNo)     %si analizza elementi di ingNo diversi da 0
    if ingNo(i)~=0        %Ti non pu\'o avere ingNO(i)
        if isempty(find(pr(inpr(i):inpr(i+1)-1)==ingNo(i)))  %se tra i
                          %possibili ingressi da dare a Ti ingNo(i) non
                          %compare allora non si attiva di certo il ciclo
            check=1;
        end
    end
end
if (check==1)&&(indy<length(y)-1)     %ciclo non si attiva e ci sono ancora
                                      %cicli da analizzare
                                      %si richiama la stessa funzione
    esCicRic(H,x,y,indy+1,pr,inpr,P)
```

```
elseif (check==0)&&(indy<length(y)-1)    %ciclo pu\'o attivarsi e ci sono
                                          %ancora cicli da analizzare
        for i=1:length(ingNo) %a turno tolgo negli ingressi possibili da Ti
                               %l'ingNo relativo
            if ingNo(i)~=0  %e chiamo ricorsione su ogni diverso
                            %gruppo di ingressi
                prric=pr;   %possibili guardando al ciclo succ
                prric(inpr(i)+ingNo(i)-1)=0;
                esCicRic(H,x,y,indy+1,prric,inpr,P);
            end
        end
elseif (check==1)&&(indy==length(y)-1) %ciclo non si attiva e sono stati
                                        %analizzati tutti i cicli
            ingok=[ingok pr];    %si aggiunge pr ottenuto a ingressi
                                 %di retraozione
            indingok=[indingok length(ingok)+1];
else     %sono stati analizzati tutti i cicli e nell'ultimo sono stati
                %riscontrati problemi
    for i=1:length(ingNo) %si scandisce tutto il vettore che contiene
                          %gli ingressi incompatibili
        check=1;
        if ingNo(i)~=0; %si toglie di volta in volta elemento problematico
            prric=pr;
            prric(inpr(i)+ingNo(i)-1)=0;      %elemento da escludere \'e tolto
            for i=1:P    %se per\'o per dato Ti non ci sono ingressi possibili
                if prric((i-1)*M+1:i*M)==zeros(1,M)
                    check=0;      %check messo a 0
                end
            end
            if check==1    %solo se per tutti i Ti esiste almeno un ingresso
                ingok=[ingok prric];      %si aggiunge pr ottenuto a ingr di
                                          %retroazione
                indingok=[indingok length(ingok)+1];
            end
        end
    end
end
end
```

Per ultima si presenta la tecnica a forza bruta, che verifica la nilpotenza delle sottomatrici.

```
%% la funzione cerca le possibili retroazioni controllando che le
```

```
% sottomatrici principali siano nilpotenti

function [ret indr]=retNilp(vL,H,N,P,M)

% per ogni Ti prende ingressi compatibili con se stesso (non devono
%  esistere cicli tra gli stati di Ti stesso)
L=creaLBCN(vL,N);     %funzione che crea la matrice in forma estesa
dT=[1];   %contiene gli indici di partenza della partizione di ogni Li
          %nelle diverse Ti
h=1;
i=1;
while i<=N
    if H(i)==h
        i=i+1;
    else
        h=h+1;
        dT=[dT i];
        i=i+1;
    end
end
dT=[dT N+1];
ingCom=[];   %conterr\'a tra indici indiC(i):indiC(i+1) gli ingressi che
             %sono compatibili controllando sottomatrice degli stati
indiC=[1];   % appartenenti a Ti
ch=0;
for i=1:M
    indL=(i-1)*N+[1:dT(2)-1];
    Ln=L(:,indL);
    check=0;
    if Ln(1,1)==1
        check=1;
    end
    if dT(2)-dT(1)>1 && check==1
        stes=[2:dT(2)-1];
        ds=length(stes);
        Ln(:,1)=[];
        Ln=Ln(stes,:);
        if Ln^ds==zeros(ds)
            ingCom=[ingCom i];
            ch=1;
        end
    elseif dT(2)-dT(1)==1 && check==1
        ch=1;
    end
```

```
end
if ch==0
    disp('Non esistono ingressi per la retroazione statica, si creano
        cicli oltre a quelli di xe in T1 per ogni possibile ingresso');
    ret=[];
    indr=[];
    return
end
indiC=[indiC length(ingCom)+1];
for i=2:P
    ch=0;
    for j=1:M
        indL=(j-1)*N+[dT(i):dT(i+1)-1];
        stes=[dT(i):dT(i+1)-1];
        ds=length(stes);
        Ln=L(:,indL);
        Ln=Ln(stes,:);
        if Ln^ds==zeros(ds)
            ingCom=[ingCom j];
            ch=1;
        end
    end
    if ch==0
        disp(['Non esistono ingressi per la retroazione statica, si
            creano cicli oltre a quelli di xe in T', num2str(i) ,' per
            ogni possibile ingresso']);
        ret=[];
        indr=[];
         return
    end
    indiC=[indiC length(ingCom)+1];
end

%a questo punto ingCom contiene gli ingressi che possono essere usati dai
%singoli Ti senza attivare loro cicli; da qui in poi si uniscono i diversi
%Ti controllando sempre che le sottmatrici diagonali principali siano
%nilpotenti

global ret; %conterr\'a gli ingressi di retroazione che stabilizzano
ret=[];

for i=1:(indiC(2)-indiC(1)) %per ogni valore di ingresso che pu\'o andare
                            %bene per T1 parte funzione
    lj=2;    %qual'\'e il Ti che va esplorato successivamente
```

```
    ContNil(vL,dT,ingCom(i),lj,N,P,ingCom,indiC);
end
if isempty(ret) %non \'e stata trovata nessuna retroazione possibile
    disp('Non␣esiste␣retroazione␣dall''uscita␣statica')
    ret=[];
    indr=[];
    return
end
indr=[1:P:length(ret)+1]; %calcolo dell'indice di ret
end

function ContNil(vL,dT,ing,lj,N,P,ingCom,indiC);
global ret;
L=creaLBCN(vL,N);
pL=[];
for i=1:lj-1      %crea la matrice Ltilde dei primi Tjl−1 bl col che \'e
                  %compatibile
    indL=(ing(i)-1)*N+[dT(i):dT(i+1)-1];
    pL=[pL L(:,indL)];
end
pL(:,1)=[]; %non appartiene a sottomatrice principale
for i=1:(indiC(lj+1)-indiC(lj))    %per tutti gli ingressi possibili per Tlj
    indL=(ingCom(indiC(lj)+i-1)-1)*N+[dT(lj):dT(lj+1)-1];    %indici relat a
                                                             %ingr i di Tlj in L
    stes=[dT(1)+1:dT(lj+1)-1];          %stati da T1−Tlj
    Li=[pL L(:,indL)];
    Li=Li(stes,:);  % sottomatrice principale
    dimLi=length(Li);    %dim della sott princ
    Li=Li^dimLi;     %potenza dimLi della sottomatrice
    if sum(sum(Li==zeros(dimLi)))==dimLi^2 && lj<P   %se risulta nilpotente
                                                     %e ho ancora uscite
        ings=[ing ingCom(indiC(lj)+i-1)];     %da trovare corrispondente ingresso
        ljs=lj+1;
        ContNil(vL,dT,ings,ljs,N,P,ingCom,indiC); %chiama la funzione ric.
    elseif sum(sum(Li==zeros(dimLi)))==dimLi^2 && lj==P       %\'e nilpotente e
                                                              %\'e stato trovato ing
        ings=[ing ingCom(indiC(lj)+i-1)];%per ogni uscita=> lavoro concluso
        ret=[ret ings];
    end       %non era nilpotente Li=>non va bene=> inutile proseguire
end
end
```

A.4 Retroazione dall'uscita tempo variante

Il codice Matlab presentato di seguito calcola una sequenza di matrici di retroazione dall'uscita per un controllo tempo variante.

```
%% la funzione cerca una sequenza di matrici per la retroazione tempo
%  variante

function [K indK t]=retTempoVar(vL,H,N,M,P)
global comp;
global Afin;

%% preparazione iniziale

% ingressi per ogni stato per passare a corona inferiore e suddivisione
% degli stati a seconda dell'uscita
comp=0;
Afin=[];
[cor Q indQ stab]=stabCorone(vL,H,N,P,M,1);
if stab==0
    disp('Ci sono degli stati da cui xe non \'e raggiungibile');
    K=[];
    indK=[];
    t=[];
    return
end
Ing=cell(N,1);
for i=1:N
    Ing{i,1}=Q(indQ(i):indQ(i+1)-1);      %ingressi inseriti in una struttura
                                          %dati pi semplice da usare (el in pos {i,1}
end                                       % sono quelli da usare per lo stato \delta_\ds^i
S=cell(P,1);      %el i-esimo contiene gli stati che danno i come uscita
C=cell(P,1);      %contiene in posizione {i,1}(1,j) la corona a cui
                  %appartiene lo stato inserito in S{i,1}(1,j)
for i=1:N    %inserisco ogni stato nel relativo Ok
    kk=H(i);      %stato fornisce come uscita kk
    S{kk,1}=[S{kk,1} i]; %aggiungo stato a Skk
    C{kk,1}=[C{kk,1} cor(i)];
end

% analisi degli stati appartenenti a S{1,1}: quelli che non possono con
% ingressi di equilibrio per xe passare a corona inferiore sono
% immagazzinati in S1str
```

```
S1str=[];
if length(S{1,1})>1      %per gli stati di S1, assicuro che gli ingressi che
                          %servono per passare da una corona
    for i=2:length(S{1,1}) %a quella inferiore siano anche di equilibrio
                           %per xe
        Qn=[];
        check=0;
        for j=1:length(Ing{1,1})
            if isempty(find(Ing{i,1}==Ing{1,1}(1,j), 1))==0
                Qn=[Qn Ing{1,1}(1,j)];   %gli ingressi da usare sono sono
                                         %quelli che permettono a xe di rimanere li
                check=1;
            end
        end
        if check==0
            S1str=[S1str i];
        end
        Ing{i,1}=Qn;
    end
end

%analisi dei cicli che coinvolgono elementi di S1str; sono quelli che
%possono creare dei problemi nello svolgersi dell'algoritmo
if isempty(S1str)==0      %nel caso ci siamno el in S1str
    vL1=vL;       %si crea BCN modificata
    for i=1:M
        if isempty(find(i==Ing{1,1}))
            for j=2:length(S{1,1})
                vL1((i-1)*N+j)=j;
            end
        end
    end
    [cor Q indQ stab]=stabCorone(vL1,H,N,P,M,1);
    if stab==0
        disp('Ci␣sono␣degli␣stati␣da␣cui␣xe␣non␣\'e␣raggiungibile');
        K=[];
        indK=[];
        t=[];
        return
    end
    Ing=cell(N,1);
    for i=1:N
        Ing{i,1}=Q(indQ(i):indQ(i+1)-1);
    end
```

```
    S=cell(P,1);
    C=cell(P,1);
    for i=1:N    %inserisco ogni stato nel relativo Ok
        kk=H(i);      %stato fornisce come uscita kk
        S{kk,1}=[S{kk,1} i]; %aggiungo stato a Skk
        C{kk,1}=[C{kk,1} cor(i)];
    end
end
Ingn=Ing;

fine=0;
K=[];    %conterr\'a tutte le matrici di retroazione nel tempo
while fine==0
    Kt=[];  %matrice di retroazione da usare al tempo t
    St1=cell(P,1);  %stati in cui si pu\'o essere al tempo t+1
    Ct1=cell(P,1);
    if length(S{1,1})>1
        x=find(C{1,1}>0);    %ingresso che vada bene per S1
        cmin=min(C{1,1}(1,x));
        indses=find(C{1,1}==cmin);    %contiene indice di uno degli stati
                                      %che appartiene a corona minore
        indses=indses(1);
        ses=S{1,1}(1,indses);
        ing=Ingn{ses,1}(1,1);      %ingresso per passare a corona inferiore
        Kt=[Kt ing];
        St1{1,1}=[1];    %stato di equilibrio ci \'e rimasto
        Ct1{1,1}=[0];
        if length(S{1,1})>1
            starr=[0 2:N]; %stati di arrivo non ancora messi
            for i=2:length(S{1,1}) %tutti gli stati di S1
                x=vL((ing-1)*N+S{1,1}(1,i));    %stato dove arrivo con
                                                %ingresso ing
                if isempty(find(starr==x, 1))==0    %non \'e stato ancora
                                                    %inserito questo stato in St1
                    usc=H(x);  %quale uscita produce
                    St1{usc,1}=[St1{usc,1} x]; %aggiunto a St1 e corona
                                               %corrispondente
                    Ct1{usc,1}=[Ct1{usc,1} cor(x)];
                    starr(x)=0;      %stato x \'e stato inserito
                end
            end
        end
    else
        Kt=[Kt Ingn{1,1}(1,1)];
```

```
        St1{1,1}=[1];
        Ct1{1,1}=[0];
        starr=[0 2:N]; %stati di arrivo non ancora messi
    end
    for i=2:P
        if isempty(S{i,1})   %si pu\'o usare qualsiasi ingresso tanto non si
                             %\'e in uno stato di questi
            Kt=[Kt 1];
        else      %bisogna individuare ingresso da usare
        indses=find(C{i,1}==min(C{i,1}));
        indses=indses(1);    %contiene indice di uno degli stati che
                             %appartiene a corona minore
        ses=S{i,1}(1,indses);
        ing=Ingn{ses,1}(1,1);      %ingresso per passare a corona inferiore
        Kt=[Kt ing];
        for j=1:length(S{i,1}) %tutti gli stati di S1
            x=vL((ing-1)*N+S{i,1}(1,j));  %stato dove arrivo con ingresso
            if isempty(find(starr==x, 1))==0    %non \'e stato ancora
                                                %inserito questo stato in St1
                usc=H(x);  %quale uscita produce
                St1{usc,1}=[St1{usc,1} x]; %aggiunto a St1 e corona
                                           %corrispondente
                Ct1{usc,1}=[Ct1{usc,1} cor(x)];
                starr(x)=0;      %stato x \'e stato inserito
            end
        end
        end
    end
    S=St1;  %aggiornamento S e relative corone
    C=Ct1;
    K=[K Kt];     %aggiunta nuova K
    check=0;      %se diventa uno il sistema \'e stato stabilizzato
    if (length(S{1,1})==1) && (S{1,1}==[1]) %S1 contiene solo 1
        check=1;
        for i=2:P    %tutti gli altri Si sono vuoti
            if isempty(S{i,1})==0
                check=0;
            end
        end
    end
    if check==1
        fine=1; %processo concluso
    end
end
```

```
indK=[1:P:length(K)+1];
t=[0:length(indK)-2];
end

%% La funzione controlla che il sistema sia stabilizzabile allo stato e
% fornisce come output la corona a cui appartiene ogni stato e anche gli
%ingressi che permettono di passare dalla corona a quella inferiore. stab
%ha valore 1 se il sistema \'e stabilizzabile, 0 altrimenti

function [corone P indP stab]=stabCorone(vL,H,N,Ping,M,e)
%controlla che BCN sia stabilizzabile a e
stab=0; %se diventa 1 il sistema \'e stabilizzabile
L=creaLBCN(vL,N);     %matrice L in forma estesa
Ltot=L(:,1:N);   %creazione di Ltot
for i=1:M-1    %per ogni ingresso aggiungo elementi a Ltot
    Ltot=Ltot|L(:,i*N+1:(i+1)*N);
end
st=[1:N];
corone=zeros(1,N);   %nella pos i conterr\'a la corona a cui appartiene lo
                     %stato delta_N^i
if Ltot(e,e)==0 %lo stato e non \'e di equilibrio=>errore
    disp('lo␣stato␣a␣cui␣si␣vuole␣stabilizzare␣non␣\'e␣di␣equilibrio
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣per␣nessun␣ingresso');
    P=[];
    indP=[];
    corone=[];
    return
else      %stato di equilibrio appartiene a corona 0
    st(e)=0;
end
Ltot=double(Ltot);
Ltotpot=Ltot;
for i=1:N-1    %calcolo potenze di Ltot per stabilire se \'e stabilizzabile
               %e divido stati in corone
    for j=1:N    %guarda le corone
        if Ltotpot(e,j)~=0 && st(j)~=0   %\'e pot i-1, se el \'e diverso da 0
            st(j)=0; %e st rel \'e 0, allora \'e prima volta che stato j ragg e
            corone(j)=i;          %e quindi appartiene a corona i
        end
    end
    Ltotpot=Ltotpot*Ltot;
    Ltotpot=Ltotpot>0; %non mi importa valore, basta che sia diverso da 0
    Ltotpot=double(Ltotpot);
end
```

```
if isempty(find(Ltotpot(e,:)==0))==0   %non \'e raggiungibile da tutti gli
                                                    %stati=>no stabilizzabilit\'a
    disp('lo␣stato␣a␣cui␣si␣vuole␣stabilizzare␣non␣\'e␣raggiungibile
␣␣␣␣␣␣␣␣da␣tutti␣gli␣stati');
    P=[];
    indP=[];
    corone=[];
    return
end
stab=1;
P=[];    %conterr\'a i Pi relativi ad ogni stato
indP=[1];      %el i contiene indice di P da cui parte Pi dello stato i
for i=1:N    %costruzione dei Pi
    cor=corone(i);   %corona a cui appartiene stato in esame
    if cor==0    %\'e stato di equilibrio
        for j=1:M %cerco quali sono ingr di equil
            if vL((j-1)*N+i)==i       %\'e ingr di equil
                P=[P j];
            end
        end
    else      %non \'e stato a cui bisogna stabilizzare
        for j=1:M %cerco quali sono ingr che portano a corona minore
            if corone(vL((j-1)*N+i))==cor-1 %se stato a cui arrivo
                                %appartiene a corona minore lo aggiungo
                P=[P j];
            end
        end
    end
    indP=[indP length(P)+1];      %aggiornamento indice P
end
end
```

A.5 Modello dell'operone *lac*

Nel seguito viene presentato il codice scritto per calcolare il modello dell'applicazione, insieme alla funzione che è stata creata per ridurre la rete agli effettivi 432 stati e alla funzione che cambia la numerazione degli stati, per portare in prima posizione lo stato a cui si vuole stabilizzare il sistema (per rispettare le ipotesi dettate all'inizio del Paragrafo 4.2.2).

```
%% costruzione della matrice L a partire dalle funzioni logiche delle
% diverse componenti
```

```
%% combinazioni ingressi

ups=[];     %ogni riga conterr\'a tutte le possibili combinazioni valide
            %per u_1,u_2,u_3
for i=0:7
    ups=[dec2bin(i,3);ups];
end
ups(2,:)=[];      %l'ingresso  1,1,0 e 0,1,0 non \'e valido, vengono eliminati
ups(5,:)=[];
% ogni riga di ups contiene dei valori per u_i validi (e l'insieme di tutte
%le righe contiene tutti i possibili valori validi)
valnodi=[];%contiene tutte le possibili combinazioni dei valori degli stati
H=[];                   %matrice contenente le uscite
statiNo=[];
for i=0:2^10-1
    valnodi=[dec2bin(i,10);valnodi];
    h=dec2bin(i,10);
   if (h(3)==1 && h(8)==0) || (h(4)==1 && h(9)==0) || (h(5)==1 && h(10)==0)
        stEl=[];
        for j=1:10
            stEl=[stEl,num2str(h(j))];
        end
        statiNo=[statiNo nod2vet(stEl,10)]; %contiene stati che non sono
   end                  %effettivamente possibili
    h1=[num2str(h(1)) num2str(h(10))];
    H=[nod2vet(h1,3) H];
%    h1=[num2str(h(1)) num2str(h(10)) num2str(h(6))];  %decommentare per
%    H=[nod2vet(h1,3) H];      %avere uscita y_2

end

numI=size(ups);
numI=numI(1);
numS=size(valnodi);
numS=numS(1);
matL=[];
for i=1:numI      %per ogni ingresso
    L_i=[];
    for j=1:numS      %per ogni diversa combinazione di stati
        G_e=ups(i,1);
        L_e=ups(i,2);
        L_em=ups(i,3);
        M=valnodi(j,1);
        B=valnodi(j,2);
```

```
        R=valnodi(j,3);
        A=valnodi(j,4);
        L=valnodi(j,5);
        P=valnodi(j,6);
        C=valnodi(j,7);
        R_m=valnodi(j,8);
        A_m=valnodi(j,9);
        L_m=valnodi(j,10);
        %aggiornamento degli stati con questi valori
        nM=C & ~R & ~R_m;
        nB=M;
        nR=~A & ~A_m;
        nA=L & B;
        nL=P & L_e & ~G_e;
        nP=M;
        nC=~G_e;
        nR_m=(~A & ~A_m) | R;
        nA_m=L | L_m;
        nL_m=((L_em & P) | L_e) & ~G_e;
        nuovoStato=[num2str(nM) num2str(nB) num2str(nR) num2str(nA)...
                    num2str(nL) num2str(nP) num2str(nC) num2str(nR_m)...
                    num2str(nA_m) num2str(nL_m)];
        L_i=[L_i nod2vet(nuovoStato,10)]; %aggiornamento Li
    end
    matL=[matL L_i];
end
L1=matL(:,1:1024);      %diversi sottosistemi
L2=matL(:,1025:1024*2);
L3=matL(:,1024*2+1:1024*3);
L4=matL(:,1024*3+1:1024*4);
L5=matL(:,1024*4+1:1024*5);
L6=matL(:,1024*5+1:1024*6);
disp('cicli␣con␣ingresso␣1')    %cicli dei sottosistemi
[x1 y1]=cicliBCN(L1,1);
disp('cicli␣con␣ingresso␣2')
[x2 y2]=cicliBCN(L2,1);
disp('cicli␣con␣ingresso␣3')
[x3 y3]=cicliBCN(L3,1);
disp('cicli␣con␣ingresso␣4')
[x4 y4]=cicliBCN(L4,1);
disp('cicli␣con␣ingresso␣5')
[x5 y5]=cicliBCN(L5,1);
disp('cicli␣con␣ingresso␣6')
[x6 y6]=cicliBCN(L6,1);
```

```
%% trasforma una combinazione di valori per i nodi della rete nell'indice i
% del corrispondente vettore di stato

function i=nod2vet(numbin,N)
i=0;
for j=1:N
    i=i+(1-str2num(numbin(j)))*2^(N-j);
end
i=i+1;
end

%% trasforma il valore della variabile vettoriale nella corrispondente
% combinazione di variabili booleane

function p=vet2nod(i,N)
q=2^N-i;
p='';
for j=1:N
    pro=floor(q/2^(N-j));
    p=[p,num2str(pro)];
    q=q-2^(N-j)*pro;
end
end

%% la funzione elimina gli stati che presentano combinazioni delle
% variabili che  non sono possibili

function [nLt ns vs nH nN]=eliminazione(L,H,N,M,statiNo)
Li=cell(M,1);    %contiene le diverse sottomatrici
nH=H;
for i=1:M
    Li{i,1}=L(:,(i-1)*N+1:i*N);
end
for i=1:length(statiNo)
    nH(:,statiNo(i))=[]; %elimino da H e da L colonne degli statiNo
    for j=1:M
        Li{j,1}(:,statiNo(i))=[];
    end
end
statin=[];
stativ=zeros(1,N);
sf=1;
for i=1:N
```

```
    if isempty(find(i==statiNo))      %nuova nominazione degli stati
        statin=[statin i];   %in posizione i il valore che aveva all'inizio
                             %stato i
        stativ(i)=sf;    %in posizione i nuovo valore dello stato i
                         %nella numerazione di partenza
        sf=sf+1;
    end
end
nLi=cell(M,1);
for i=1:M    %aggiornamento valori di L
    for j=1:length(Li{i,1})
        sa=Li{i,1}(1,j);
        nLi{i,1}=[nLi{i,1} stativ(sa)];
    end
end
nLt=[];
for i=1:M
    nLt=[nLt nLi{i,1}]; %matrice L totale
end
ns=statin;
vs=stativ;
nN=length(Li{1,1});
end

%% la funzione rinomina gli stati in modo che rispondano alle ipotesi
%% introdotte nel paragrafo della retroazione dall'uscita tempo invariante

function [nLt ns vs nH trH]=rinominazione(L,H,xe,N,P,M)
he=H(xe);    %uscita dello stato a cui si vuole stabilizzare
stn=cell(P,1);   %in posizione {i,1} contiene gli stati che danno come
stn{he,1}=[xe]; %uscita i
for i=1:N
    if i~=xe
        hi=H(i);
        stn{hi,1}=[stn{hi,1} i];
    end
end
ns=stn{he,1};
nH=[he];
trH=[he];
for i=1:P
    if i~=he
        ns=[ns stn{i,1}]; %in posizione i c'\'e la vecchia numerazione di
                          %quello che adesso \'e lo stato i
```

```
        trH=[trH i];      %l'uscita i \'e la vecchia uscita trH(i)
    end
end
nH=[];
usc=1;
for i=1:length(stn{he,1})
    nH=[nH usc];      %costruzione della nuova matrice di uscita
end
for i=1:P
    if i~=he
        usc=usc+1;
        for j=1:length(stn{i,1})
            nH=[nH usc];
        end
    end
end
vs=zeros(1,N);
for i=1:N
   vs(ns(i))=i;%in posizione i di vs c'\'e la numerazione nuova dello stato i
end
nL=cell(M,1);    %sottomatrici della rete rinominata
for j=1:M
    Lj=L((j-1)*N+1:j*N);
    for i=1:length(Lj)
        chi=ns(i);
        dove=Lj(chi);
        nchi=vs(dove);
        nL{j,1}=[nL{j,1} nchi];
    end
end
nLt=[]; %nuova matrice L totale
for i=1:M
    nLt=[nLt nL{i,1}];
end
end
```

Ringraziamenti

Con questa tesi si conclude un percorso di studi cominciato 5 lunghi anni addietro, anzi, per meglio dire, quasi 20 anni fa; se sono arrivata fin qua, è anche merito di bellissime persone che ho incontrato in questo percorso (ma anche al di fuori di esso), che mi hanno a loro modo sempre aiutato. Rubo quindi un pó di tempo alla stesura della tesi (che spero ormai sia a buon punto), per ringraziare le persone che mi stanno a cuore (sperando che anche tu che stai leggendo questa parte riesca a trovare una parte a te dedicata; in caso contrario, non averne a male, si sa che son cialtrona di natura, no??).

Un primo gran ringraziamento va al Professor **Fornasini**, che mi ha fatto adorare Teoria dei Sistemi e tutti i suoi derivati; è un bravissimo Professore, e per la tesi mi ha pazientemente seguito (interessandosi anche al corso di cucina), ed è sempre stato molto disponibile.
Un secondo ringraziamento va alla Professoressa **Ronconi**, che mi ha trasmesso l'amore per l'Algebra Lineare, e mi ha seguito amorevolmente per la tesi triennale.
Per quanto riguarda questa tesi, un ringraziamento è doveroso anche alla Professoressa **Valcher**, che è stata molto disponibile e mi ha fornito utili suggerimenti.
A seguire si ringraziano tutti i Professori e gli insegnanti che ho avuto, e in particolare il Prof. **Marson**, che ha reso comprensibile un'ostica Analisi 2, e la Prof.a **Marsullo** delle superiori, che, anche grazie alla paura che mi incuteva, mi ha fatto imparare bene l'inglese.

Uscendo dal mondo scolastico, i primi che voglio ringraziare sono decisamente i miei genitori, che mi hanno sempre sostenuto e incoraggiato in tutto: la **mamma**, che in tutti questi anni ha sopportato numerose crisi isteriche di varia natura, e il **babbo**, che pur facendomi molte volte arrabbiare (alle volte ammetiamolo, mi arrabbio io gratuitamente), è in realtà sempre pronto ad aiutarmi. Segue il mio **fratellone maggiore**, che ha sempre una parola buona per me, ed è una delle persone più buone e generose che io conosca, e poi il **fratello medio**, che mi ha tanto stressato da piccola, ma con il quale ora condividiamo l'amore per i Los.
I restanti **parenti** li ringrazio tutti assieme, sapendo che sostengono le mie idee balzane su quello che farò nella vita.
Ringrazio qui anche la mia **Francy**, poichè, se sono cresciuta così, è anche merito suo (e del suo "Varda che se te caschi e te te fè mae, te dago anca!")!

E arriviamo agli amici: ne figurano di tutti i tipi, presenti da più o meno tutta la vita, ma anche di recente acquisizione, persone pacate da cui non ti aspetti una pazzia a pagarla oro, e persone pazze che quando son tranquille incutono paura. A qualunque tipologia voi apparteniate, è assolutamente indifferente, c'è un posticino per ognuno di voi nel mio cuore!
Apriamo le danze, con l'avviso che l'ordine è misto, dai più vecchi ai più nuovi, e non è sempre detto...
In prima fila partono le amiche che ho sempre avuto accanto, o che comunque mi sono state accanto per molto molto tempo:
un ringraziamento a **sorela de Paola**: il kilometro e 700 metri che ci separano (ringraziamo anche Google Maps) risulta una distanza inaffrontabile per noi due, tolta quella volta ogni 3 mesi in cui si decide che c'è proprio la necessità. Nonostante questo, sappiamo che non servono sempre molte parole e incontri per capirsi nel profondo no? La tua forza di volontà è sempre un grande esempio, tienilo a mente!
Uno a **sorela de Mela**, con cui la distanza è leggermente maggiore, ma la sostanza non cambia...ci si vede poco, ma si finisce sempre per ridere e ricordare vecchi tempi spensierati!
Ed eccone uno per la mia cara **Helen**: grazie, perché non mi rifiuti mai un buonissimo tè e perché sei sempre amorevolmente pronta ad offrirmi un pompelmo, e devo anche ringraziarti per un utilissimo libro di biologia! Sei proprio una amica del cuore (evito la e solo per decenza, ma non crediate che siano smancerie queste...).
Infine uno ad un'altra grande donna: a te **Gloria**, perché la calma e il sangue freddo che abbiamo sviluppato in 5 anni assieme, me li sono sempre portati dietro ad ogni esame...sei fantastica!

Ed ora via con i **Leosi**: a **Borto**, che è la pazienza e l'organizzazione fatta persona: sei di titanica importanza, per quanto tutti ti dicano su e tu sia il capro espiatorio del gruppo; a **Bert**, che si preoccupa di farci rispettare la natura, da buon ecologista, e che da buon salutista non usa molto sale e copre il pancino; a **Pep**, il mio psicologo di fiducia, per il suo essere un diamante inscalfibile, nulla può toccarti; a **Furio**, perché tu sei lo sbarazzino e lo spensierato, ed hai un cuore d'oro; a **AleP**, la donna che ha sempre il sorriso pronto e la voglia di fare qualcosa di insensato; a **AleG** per i suoi riccioli sbarazzini e per darmi soddisfazione mangiando tanti grissini; a **Sancho**, per aver molte volte tentato di istruirmi in Geografia, seppure con scarsi risultati, e per tutti i piccoli aneddoti delle superiori; a **Rachele**, la nostra Bordina di fiducia, nonché designer

e creativa, e soprattutto mastra salsicciaia; a **Zeno**, l'uomo dal fascino tenebroso che suona in banda, il cui fascino sta scomparendo perché non si fa più vedere in giro.

E si arriva quindi all'ingegneristico **All Dei Long** (mamma come siamo nerdosi): a **Bedo**, che ha pazientemente coordinato i movimenti della barca, prima come nostromo e poi come timoniere...chissà quante volte sarei finita su una secca senza di te...conserverò i quaderni con le nostre cavolate a vita, e brinderò alla tua salute con tante medie; a **Baruz**, perché sono convinta che Batman mi protegge ovunque vado, e perché mi ha fatto sentire una vera nerd comandando il Cluster del Dei da casa; ad **Emma**, perché è sempre piena di vita e di ritmo, e a **Martina**, per la tua creatività e voglia di fare (siete insieme perché insieme facciamo l'ancora di salvezza per tutti gli uomini del gruppo, che altrimenti parlerebbero solo di programmi e cose tecnologiche, e in più perché vi devo ringraziare per un acronimo, che penso vi ricorderà qualcosa...SAD); a **Vanuz**, il biondo numero uno del gruppo, il mio PR di fiducia che non molla mai, nonché portatore di fazzoletti con una eleganza che solo lui ha, per esaltare un suo lato che lo ha reso molto famoso; ad **Ale**, il mio figliolo emigrato, che ha preso dalla mamma la passione della cucina...continua così; a **Giovi**, perché è un vero uomo di stile, ed è in grado di fornire ottimi consigli in campo amoroso; a **Fede**, il mio fratello...non ci sono parole per esprimere la nostra affinità, possiamo solo dire di essere stati separati dalla nascita; infine arriviamo al biondo numero 2, **Colla**, che non si fa mettere i piedi in testa da Vanuz, e che trova sempre una duplice lettura ad ogni affermazione.

Visto che di Università stiamo parlando, ringrazio tutti i miei compagni, in triennale e in magistrale: ho passato ore molto divertenti, ed è stato bello condividere insieme anche le sudate carte; in particolare i ringraziamenti vanno a **Lele**, il sultano, per avermi fatto compagnia innumerevoli volte in mensa; a **Marco R.**, l'esteta, per le risate assicurate e per avere gran riflessi; ad **Ombre**, per saper ascoltare, e per i pranzi offerti; a **Luca F.** per tutte le ore passate insieme l'ultimo semestre, e per aver giuocato a non so che giuoco; a **Fabio Laa-Laa**, perché vederlo in Da rendeva la giornata più simpatica e colorata; a **Francesca**, per avere sempre il sorriso pronto, e sempre tanto entusiasmo; a **Tano** per il suo entusiasmo contagioso e per le ore condivise a studiare insieme ISD; a **Martina** per la sua serietà durante i gruppi di studio per S&F; a **Michele** per essere pieno di vita e ottimista; a **Giulio M.** per la sua grande forza d'animo (sei un esempio); al **Terrore**, per aver condiviso

con me l'esperienza della pizza Kebab; a **Gioia**, con la quale ci inseguiremo probabilmente a vita, ma forse troveremo il tempo per un caffè; a **Frack**, perchè è Frack, e perchè regge l'alcool come un vero maestro.

Proseguiamo con un fritto misto, perchè alcuni rientrano in categorie miste: alla **Ele**, per le grandi risate del primo anno con Gianni e Pinotto, e per avermi assecondato nel fare la macchina lungo il Piovego; a **Pone**, perché è un buon figliolo e perché custodisce con amore Poldo; a **Kimi**, l'uomo sicuro di sè, per avermi scroccato numerosi dolci e per darmi uno scopo nella vita; a **Giulio C.**, per le tue innumerevoli perle di saggezza, e per avermi fatto riconoscere Zucchetta, il procione assassino e l'unicorno a due corna; a **Gotty**, per le discussioni su Tiziano Ferro; a **Bildi**, perché il sabato sera non asseconda sempre Vanuz e non gli permette di bere; alla **Lucy**, per la sua ospitalità, e perché, pur essendo piccola, è un concentrato di energia...non mollare!

Un ringraziamento al mio vicino di casa preferito, **Lorenzo**, per avermi fatto conoscere Zoolander: è una pietra miliare della mia vita quel film; uno al **Castagnino medio**, ed in particolare ai suoi capelli sbarazzini e adorabili; un ringraziamento grande anche a **Popa**, **Seba** e **Pack**, per le tante fatiche condivise, ma soprattutto per le risate e le cavolate vissute assieme. Altro enorme ringraziamento a **Ricci**, **Christian** e **Dario**, che hanno vegliato sulla mia crescita come fratelli, e mi hanno dato preziosi consigli, in particolare su come si usano i libri. Un ringraziamento ad **Andrea**, per il suo essere una persona buona, e perché quando suona il pianoforte mi lascia sempre senza parole; uno alla **Eleonora** che apprezza i miei stessi film e uno a **Manuel**, che mi fa sempre tanto ridere con le sue fantastiche uscite; a **Giacomo**, perché non so se alla fine si ritenga invitato o meno; ad **Alexandra** che è una splendida conoscenza frutto di una magnifica vacanza in Austria; ad **Enrica**, perché è una bellissima persona, anche se è ormai una vita che non ci vediamo; a **Greta**, alla **Fra**, ad **Ale** e a tutti i **compagni del corso di cucina**, perché siete veramente delle persone simpatiche e interessanti e mi avete messo a contatto con un mondo diverso dalla mia normalità; un enorme grazie a **Martino**, perché se io già amo la cucina di mio, lui è in grado di farmela amare ancora di più: sei veramente bravissimo.

Ultimo ringraziamento, ma non meno importante, ai **Los Massadores**: non leggerete mai questo trafiletto, ma sappiate che mi avete restituito entusiasmo, gioia, voglia di fare e mi avete reso una persona ancora più sbarazzina!

Riferimenti bibliografici

[1] D. Cheng, H. Qi, Z. Li, *Analysis and Control of Boolean Networks*, Springer (2011).

[2] G. Hochma, M. Margaliot, E. Fornasini, M. E. Valcher, *Symbolic dynamics of Boolean control networks*, Automatica vol 49, pp 2525-2530 (2013).

[3] E. Fornasini, M. E. Valcher, *Observability, Reconstructability and State Observers of Boolean Control Networks*, IEEE Trans. on Automatic Control, vol. 58, pp 1390-1401 (2013).

[4] E. Fornasini, M. E. Valcher, *On the Periodic Trajectories of Boolean Control Networks*, Automatica vol. 49, pp 1506-1509 (2013).

[5] R. Li, M. Yang, T. Chu , *State Feedback Stabilization for Boolean Control Networks*, IEEE Trans. on Automatic Control, vol. 58, no. 7, pp 1853-1857 (2013).

[6] H. Li, Y. Wang, *Output Feedback Stabilization Control Design fo Boolean Control Networks*, Automatica (2013), http://dx.doi.org/10.1016/j.automatica.2013.09.023.

[7] E. Fornasini, M. E. Valcher, *Optimal control of Boolean control networks*, IEEE Trans. on Automatic Control, in revisione

[8] D. Lschov, M. Margaliot, *A maximum principle for single-input Boolean Control Networks*, IEEE Trans. on Automatic Control, vol. 56, no. 4, pp 913-917 (2011).

[9] D. Lschov, M. Margaliot, *A pontryagin maximum principle for multi-input Boolean Control Networks*, Recent Advances in Dynamics and Control of Neural Networks Networks, (E. Kaslik and S. Sivasundaram, Eds.), (2011).

[10] Y. Zaho, Z. Li and D. Cheng, *Optimal Control of Logical Control Networks*, IEEE Trans. on Automatic Control, vol. 56, no. 8, pp 1766-1776 (2011).

[11] S. A. Kauffman, *Metabolic stability and epigenesis in randomly constructed genetic nets*, J. Theoretical Biology, vol. 22, pp. 437-467 (1969).

[12] A. Veliz-Cuba, B. Stigler, *Boolean Models Can Explain Bistability in the lac Operon*, Journal of Computational Biology, vol. 18, no. 6, pp. 783-794 (2011).

[13] F. Jacobs, J. Monod, *Genetic regulatory mechanisms in the synthesis of proteins*, J. Mol. Biol., vol 3, pp. 318-356 (1961).

[14] B: Goodwin, *Temporal Organization in Cells*, Academic Press0 New york (1963).

[15] M. Santillán, M. C. Mackey, *Influence of Catabolite Repression and Inducer Exclusion on the Bistable Behavior of the lac Operon*, Biophysical Journal, vol. 86, pp. 1282-1292 (2004).

[16] M. Santillán, M. C. Mackey, E. S. Zeron, *Origin of Bistability in the lac Operon*, Biophysical Journal, vol. 92, pp. 3830-3842 (2007).

[17] M. Santillán, *Bistable Behavior in a Model of the lac Operon in Escherichia coli with Variable Growth Rate*, Biophysical Journal, vol. 94, pp. 2065-2081 (2008).

[18] F. Amaldi, P. Benedetti, G. Pesole, P. Plevani, *Biologia molecolare*, Casa Editrice Ambrosiana (2011).

[19] B. M. Hogema, J. C. Arents et altri, *Inducer exclusion by glucose 6-phosphate in Escherichia coli*, Molecular Microbiology, vol.28, no. 4, pp. 755-765 (1998).

[20] A. Fauré , A. Naldi, C. Chaouiya, Denis T., *Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle*, Bioinformatics, vol. 22, no. 14, pp. 124-131 (2006).

[21] S. Bornholdt, *Boolean network models of cellular regulation: prospects and limitations*, J. R. Soc. Interface, vol. 5, no. Suppl 1, pp. 85-94 (2008).

[22] M. I. Davidich, S. Bornholdt, *Boolean Network Model Predicts Cell Cycle Sequence of Fission Yeast*, PLoS ONE, vol. 3, no. 2 (2008).

[23] F. Li, T. Long, Y. Lu, Q. Ouyang, C. Tang, *The yeast cell-cycle network is robustly designed*, PNAS, vol. 101, no. 14, pp. 4781-4786 (2004).

[24] R. Albert, H. G. Othmer, *The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in Drosophila melanogaster*, J. of Theoretical Biology, vol. 223, pp. 1-18 (2003).

[25] C. Espinosa-Soto, P. Padilla-Longoria, Elena R. Alvarez-Buylla, *A Gene Regulatory Network Model for Cell-Fate Determination during Arabidopsis thaliana Flower Development That Is Robust and Recovers Experimental Gene Expression Proles*, The Plant Cell, vol.16, pp.2923-2939 (2004).

[26] Song Li, S. M. Assmann, R. Albert, *Predicting Essential Components of Signal Transduction Networks: A Dynamic Model of Guard Cell Abscisic Acid Signaling*, vol.4, no. 10, pp. 1732-1748 (2006).

[27] M. Fischetti, *Lezioni di Ricerca Operativa*, Edizioni Libreria Progetto Padova (1999).

[28] E. Fornasini, *Appunti di teoria dei sistemi*, Edizioni Libreria Progetto Padova (2011).